3D Livewire and Live-Vessel: Minimal Path Methods for Interactive Medical Image Segmentation

by

Miranda Poon

B.A. Sc., The University of British Columbia, 2005

A THESIS SUBMITED IN PARTIAL FULFILLMENT OF
THE REQUIREMENTS FOR THE DEGREE OF

Master of Applied Science

in

THE FACULTY OF GRADUATE STUDIES

(Electrical & Computer Engineering)

THE UNIVERSITY OF BRITISH COLUMBIA
(VANCOUVER)

April 2008

Abstract

Medical image analysis is a ubiquitous and essential part of modern health care. A crucial first step to this is segmentation, which is often complicated by many factors including subject diversity, pathology, noise corruption, and poor image resolution. Traditionally, manual tracing by experts was done. While considered accurate, this process is time consuming and tedious, especially when performed slice-by-slice on three-dimensional (3D) images over large datasets or on two-dimensional (2D) but topologically complicated images such as a retinography. On the other hand, fully-automated methods are typically faster, but work best with data-dependent, carefully tuned parameters and still require user validation and refinement.

This thesis contributes to the field of medical image segmentation by proposing a highly-automated, interactive approach that effectively merges user knowledge and efficient computing. To this end, our work focuses on graph-based methods and offer globally optimal solutions. First, we present a novel method for 3D segmentation based on a 3D Livewire approach. This approach is an extension of the 2D Livewire framework, and this method is capable of handling objects with large protrusions, concavities, branching, and complex arbitrary topologies. Second, we propose a method for efficiently segmenting 2D vascular networks, called 'Live-Vessel'. Live-Vessel simultaneously extracts vessel centrelines and boundary points, and globally optimizes over both spatial variables and vessel radius. Both of our proposed methods are validated on synthetic data, real medical data, and are shown to be highly reproducible, accurate,

and efficient. Also, they were shown to be resilient to high amounts of noise and insensitive to internal parameterization.

Table of Contents

List of Tables

List of Figures

Acknowledgements

I would like to thank my supervisors, Dr. Rafeef Abugharbieh and Dr. Ghassan Hamarneh, for their guidance. I would also like to acknowledge Dr. Martin McKeown for providing T1 MRI data and the U.S. National Library of Medicine for the publicly available Visible Human Project CT data. Moreover, I would like to acknowledge M. Niemeijer for the DRIVE retinal image database. Lastly, I would like to thank the lab members of BiSICL for their expertise and support.

MIRANDA POON

The University of British Columbia

April 2008

Chapter 1

Introduction and Background

Today, medical imaging is ubiquitous and essential for medical professionals in patient diagnosis, treatment planning, and computer-aided surgery. Since most imaging techniques are non-invasive, the study of anatomical structure and disease progression *in vivo* is possible. Some examples of three-dimensional (3D) medical imaging are computer tomography (CT) to detect calcified tissue, magnetic resonance imaging (MRI) for imaging soft tissue, and magnetic resonance angiography (MRA) for capturing vasculature. Two-dimensional (2D) medical imaging includes X-ray (what CT is based on), and retinography which captures the blood vessels in the eye.

1.1 Problem Statement and Motivation

In almost all medical image applications, segmentation is a vital precursor step. Image segmentation in the broadest sense is the division and classification of an image into areas or regions of interest. However, unlike mainstream multimedia graphics, medical imaging has relatively poor image quality due to low resolution and high noise, which poses problems in designing effective segmentation algorithms. Also, since medical imaging is often used to study disease across many patients, biological diversity and pathology hinders algorithm robustness and accuracy. Furthermore, segmentation of 3D objects over large datasets raises design challenges to algorithm extensibility and computational speed. These problems have fuelled interest in finding alternatives to the predominant segmentation choices today: the slow but accurate manual tracing approach and the usually faster but inflexible fully-automated approaches.

The impact of developing successful segmentation algorithms is extremely large; thus much research in medical image processing is focused in this area. An obvious motivation is that a region of interest (ROI) can be visualized more clearly after segmentation than with the image background. Also, in 3D, embedded objects can only be visualized as a whole after segmentation, rather than viewed slice-by-slice. Segmentation is also necessary for volume and shape analysis in establishing statistical norms and disease research, for instance, in Parkinson's disease [1] and Alzheimer's disease [2] research, both of which are done on 3D data. Vessel segmentation from medical imaging is also important in studies of brain tumour vasculature [3], intracranial aneurysms [4], and diagnoses and research of diabetes and hypertension [5][6].

Moreover, both rigid body registration and deformable registration techniques benefit from segmentation, in localizing the region of interest to reduce the high computation complexity as well as increasing the accuracy of registration [7].

1.2 Overview of Image Segmentation Approaches

Medical image segmentation techniques are extremely varied. This section focuses on the benefits and drawbacks of manual, automatic, and semi-automatic techniques.

1.2.1 Manual Segmentation

Manual segmentation is the technique that requires the user to specify each image unit (pixel or voxel) as foreground or background. This is typically done using a mouse or a graphics tablet. The actual delineation process varies from manually 'painting' each pixel in the object of interest to tracing the segmentation contour around the object by an 'expert' – a user familiar to the object being segmented. Since manual segmentation is often considered accurate, it can be used to 'train' the parameters of automatic segmentation techniques (Section 1.2.2). However, while manual segmentation results are often used as the 'gold standard' for benchmarking other techniques' accuracy, they still suffer from inter and intra-subject variability [8] and user fatigue [9]. Also, for large volumes and large patient datasets, manual segmentation is highly tedious and time consuming because careful delineation is required for each 2D slice within a volume. Moreover, manually extracted contours viewed from an orthogonal direction typically appear jagged because boundary smoothness is not enforced between slices. Lastly,

oftentimes a user may not be able to see structural features due to low image quality or poor computer displays.

1.2.2 Fully-automated Segmentation Techniques

Fully-automated segmentation techniques, on the other side of the segmentation methods spectrum, offer faster segmentation task times and eliminate inter-operator variability. However, they work best when their parameters are carefully tuned for specific image properties and anatomy, which remains a challenge, and still require user validation and refinement. An additional complication factor for these approaches is that anatomical structures are typically affected by significant variations due to subject diversity and pathology which reduces the segmentation accuracy, robustness, and consistency between volumes. Under these conditions, an additional design difficulty is locating and identifying the object(s) of interest without user guidance. Additional literature review pertaining to automatic 3D segmentation and vessel segmentation are presented in Section 2.1 and Section 3.1 respectively.

One of the most basic forms of segmentation is global or adaptive thresholding, which assigns a binary value to each pixel on the image or sub-image, depending on whether the pixel's intensity is above/below a threshold value [10]. While simple and effective on certain imaging modalities (i.e. segmenting bone from CT), this method does not enforce tissue connectivity and performs poorly noisy environments.

Another segmentation group called region growing uses image intensity information and enforces connectivity of segmented pixels. This method is seeded as a single point, and a region propagates from this point until stopping conditions are met

based on image intensity change or statistical properties of the image [11]. A drawback of this property is its tendency to 'leak' outside the region of interest since it is not as robust to noise as more sophisticated techniques such as deformable models (explained later). Similar to region growing are watershed techniques which view a 2D image in 3D, with the third dimension being image intensity [10]. Regions with low pixel intensity are 'flooded' and as basins begin to meet at higher intensity levels, 'dams' are constructed, which translates into boundary paths between regions. This method suffers from the same leakage problem as region growing.

The deformable models family of segmentation techniques is much more robust than global image operators such as in the above. The goal of these techniques is to evolve a closed contour to minimize its energy function, as defined by external forces derived from image properties such as image intensity and internal forces such as contour smoothness [12]. The advantages to deformable models are that for any particular contour, pixel connectivity is ensured, and an object's boundaries can be defined even if there is not enough gradient information in the region. However, they tend to settle into local minima [12], are generally quite computational, are unpredictable during contour evolution, and may not produce accurate results without user initialization or post-correction. The two main classes in this family are explicit models which evolve explicitly defined contours [12][13][14] and implicit curves[15][16][17]. The main difference is that while explicit models are based on parametric formulations during deformation and less computational, implicit models represent the contours as a level set of higher-dimensional scalar function and can handle topological changes to the object.

1.2.3 Semi-automated Interactive Techniques

Due to the above-mentioned difficulties with both manual and fully automated segmentation techniques, semi-automated methods have drawn wide interest as a way to facilitate computer-based segmentation of 3D anatomical structures using minimal human interaction. Interaction in image segmentation typically involves parameter selection, menu operation, or graphical input. Interactive segmentation types, similar to segmentation techniques in general, are extremely varied; for in-depth surveys of existing techniques, we refer readers to [9][18]. Also, additional literature reviews specifically pertaining to 3D interactive segmentation and semi-automated vessel segmentation are presented in Section 2.1 and Section 3.1 respectively.

Tweaking parameter values, the user is given rapid feedback on the segmentation result, which is derived from a largely 'black box' algorithm in the tool's backend. Example parameter values include weighting for a deformable model [19], threshold parameters [20], and number of processing iterations [20]. This interaction scheme is simple to implement, but oftentimes the user has to have a passing knowledge of the computation algorithm and providing intuitive interaction is a design challenge [9]. However, if real-time feedback is achieved, algorithm understanding becomes less of an issue, as the user can learn through experience.

Menu-based techniques place the largest emphasis on the computational algorithm. The user is guided through the segmentation process by providing descriptive instructions regarding the object's properties using buttons and forms but otherwise does not have direct control over the segmentation contour or parameterization. Numerous

methods use this scheme to allow users to accept/reject automatically computed segmentation [21], or adopt this as part of the overall interaction task [22]. Other methods use menus in a flowchart manner to let users narrow down the ROI's topological properties [23] or to let users choose templates [24]. While results tend to be more reproducible and efficient, accuracy is not guaranteed and if the menu system is not able describe the object exactly, program robustness suffers.

Lastly, graphical input techniques require the user to specify seedpoints or manipulate artificial objects on top of the image itself, which bootstraps the computation algorithm. With direct user control over segmentation regions, robustness is maximized. This interaction scheme tends to be the most intuitive, but its design challenges are to require only minimal interaction and to minimize user-input variability. There are 3 main types of interactivity. The first is initiation of a deformable model [12][25] and graphically specifying regions for the model to favour/avoid in real-time [13][14]. The second type is specification of background and foreground seeds for computation of a quick, but automatically determined segmentation [22][23]. Lastly, the third type is specification of seedpoints along the contour itself, whether it is the object edge [26] or a tubular structure's medial path [27]. The focus of this thesis is this third type, and we explore the graph theory optimal path finding involved in the next section.

1.3 Graph-Based Segmentation

Images are raster images, meaning that they are represented by pixels and voxels, each represented by an intensity value (greyscale or RGB). While pixels and voxels are not necessarily squares and cubes, they are tiled as an array and can be represented as a

graph or matrix. A graph is a collection of nodes and the connections that each node has with other nodes. Since segmentation contours are essentially paths along the graph and delineates a sub-graph (the object or region of interest), graph-theory and graph-based approaches are useful and intuitive methods. While there are numerous graph-based segmentation approaches such as region growing [11] or Graph-Cuts based methods [22][28][29], we focus on optimal *path finding* approaches.

1.3.1 Dynamic Programming

Dynamic programming [30] is a method of solving problems which have incremental, optimal substructure on a graph, and applies to many fields including mathematics, computer science, and economics. Using dynamic programming, the globally optimal path on a graph between arbitrary nodes p to q is found by recursively breaking the problem into sub-problems and finding the optimal path for these simpler cases. In the first step, the lowest cost path to q from all adjacent nodes is determined. However, while the *local* path costs from these nodes to q are known, this algorithm is recursively applied to each node that is connected to q to find the globally optimal path and associated cumulative cost. The result of this recursion is that not only is the globally optimal path from p to q computed, but the globally optimal paths from p to all nodes visited during recursion are found as well.

An example of solving a sub-problem is illustrated in Figure 1, where the globally optimal (lowest cost) path from end node p to nodes 1, 2, and 3 were found in earlier iterations to have costs 8, 5, and 6 respectively. Since the path segment from node 1 to q

exhibits the lowest total cost (8+1), this segment becomes part of the globally optimal path.

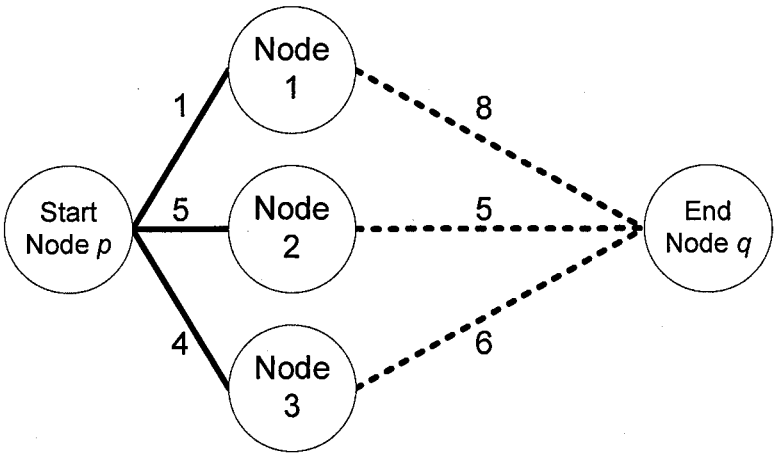


Figure 1 Although the path from Node 1 to the end node *q* has the highest optimal cost compared to Nodes 2 and 3, it has the lowest cost to the starting node; therefore, the optimal path from the start node *p* to *q* passes through node 1 instead.

1.3.2 Dijkstra's Algorithm

Dijkstra's algorithm [31] is a graph search algorithm that uses the concepts of dynamic programming to find the globally optimal (lowest cumulative path cost) path between a node and every other node on the graph. It is assumed that each node has an associated local cost and/or a path cost between it and each of its neighbours. However, its main difference to traditional dynamic programming is that it uses iteration rather than recursion. In summary, the algorithm is outlined as follows and illustrated in Figure 2:

1. Create a cumulative cost list, visited list, and a node queue list. These lists' elements are initialized as ∞, 0, and 0 respectively. The starting node has a cumulative cost of 0, is set to 'visited', and is placed on the node queue.

2. The node in the queue with the smallest cumulative cost is set to 'visited' and processed. For each of this node's unvisited neighbours, the cumulative cost is calculated based on this new path. If this cost is smaller than the previously calculated cost, it is updated in the list. If this neighbour node is not on the node queue, it is inserted into the queue.

3. The node queue is resorted according to each node's cumulative cost.

4. Steps 2 and 3 are repeated until all nodes are processed.

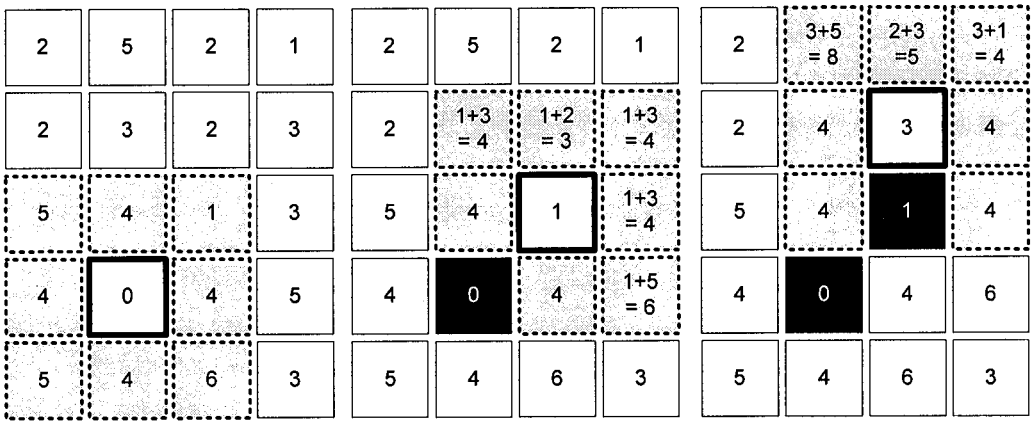


Figure 2. In the graph search of Dijkstra's algorithm, the cumulative costs of neighboring nodes (dotted squares) of the 'unvisited' node with lowest cumulative cost (bold square) are updated. Costs of 'visited' nodes (solid black squares) are no longer updated.

Using the above graph search algorithm, the final cumulative cost list is a data structure that stores both the globally optimal cumulative cost and the globally optimal path from a node to all nodes on the graph. The exact path from a starting node to a destination node is determined backwards. Starting at the destination node, the 'next' node along the path is defined as the neighbour with the lowest cumulative cost. This process is repeated until the starting node is reached. Figure 3 illustrates this process.

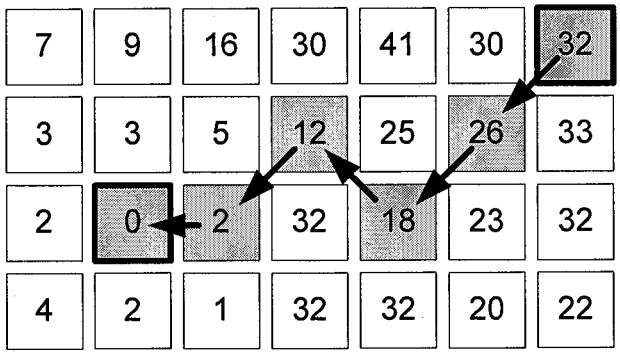


Figure 3. In iteratively determining the optimal path from a node with cumulative cost of 32, the next node on the path is always the neighboring node with the lowest cumulative cost until the seedpoint (cumulative cost of 0) is reached.

A similar method is the Fast March method [32]. This is based on Dikjstra's efficient one-pass algorithm (each node is visited at most only once), and is used to solve problems in continuous space because of its ability to interpolate between nodes. However, for discrete path-minimizing problems in a discrete space, Dijkstra's algorithm provides a more direct solution than Fast March.

1.4 The Livewire Formulation

Since segmentation contours are assumed to be continuous along the outside of the region of interest, graph-based techniques such as Dijkstra's algorithm can be easily applied. The Livewire framework [26] incorporates Dikjstra's algorithm and interactivity for real-time user-guided segmentation. Here, path optimization is based on edge detection measures, and real-time contour feedback is provided by taking advantage of the computationally inexpensive path determination step of Dijkstra's algorithm.

1.4.1 Contour Optimization Process

Ideal segmentation contours lie along the boundaries of objects, which typically exhibit high gradient. Therefore, edge detection is used to assign low local cost to edge pixels. Initially, only Laplacian of Gaussian (LoG) edge detection [10] was used, but later implementations used multiple algorithms including Canny edge detection [33] and image gradient for higher algorithm robustness. Given that $q = (x, y)$ is a pixel on the image S and $p = (x', y')$ is a neighbouring pixel of q, the Laplacian of Gaussian (LoG) cost $C_{LoG}(q)$ is defined as

$$C_{LoG}(q) = 1 - (LoG_{\text{ker}nel}(x, y) * S) \quad \Big|_{(x,y)=q} \quad ,$$

where S is convoluted with the LoG kernel

$$LoG_{\text{ker}nel}(x, y) = -\frac{1}{\pi\sigma^4}\left(1 - \frac{x^2 + y^2}{2\sigma^2}\right)\exp^{-\frac{x^2+y^2}{x\sigma^2}} \quad .$$

The gradient magnitude cost $C_G(q)$ is defined as

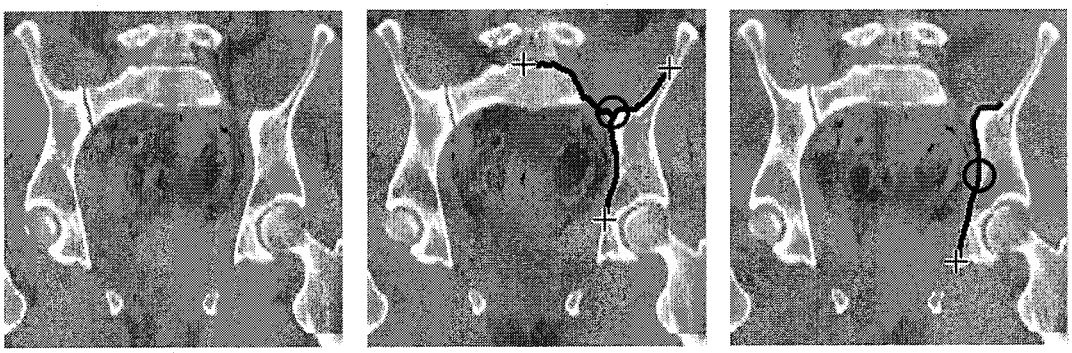
$$C_G(q) = 1 - \frac{1}{\max(G)}\sqrt{\left(\frac{dS(x,y)}{dx}\right)^2 + \left(\frac{dS(x,y)}{dy}\right)^2} \quad \Bigg|_{(x,y)=q} \quad ,$$

where max(G) denotes the largest gradient magnitude found in the image. Since object edges are smooth, the gradient direction cost $C_{GD}(p, q)$ favours pixels that show small differences in the direction of the gradient. This is defined as

$$C_{GD}(p,q) = \frac{1}{\pi}\arccos\left(\frac{S'(p) \bullet S'(q)}{G(p)G(p)}\right) ,$$

where *G(p)* and *G(q)* denote the gradient magnitude (not gradient cost) of the image at pixel p and q respectively. Each of these filters has unique strengths [10], and their aggregation results in a more robust algorithm. For instance, the LoG cost is less sensitive to image noise due to its convolution with a Gaussian kernel, while the gradient magnitude cost is more sensitive in detecting weak structural edges. Lastly, a scalar cost $C_d(p,q)$ that proportional to Euclidean distance $\sqrt{(x-x')^2+(y-y')^2}$ is added in between each node and its neighbours in order for the optimization to favour physically shorter and therefore smoother contours.

1.4.2 Interactive Seeding



(a) (b) (c)

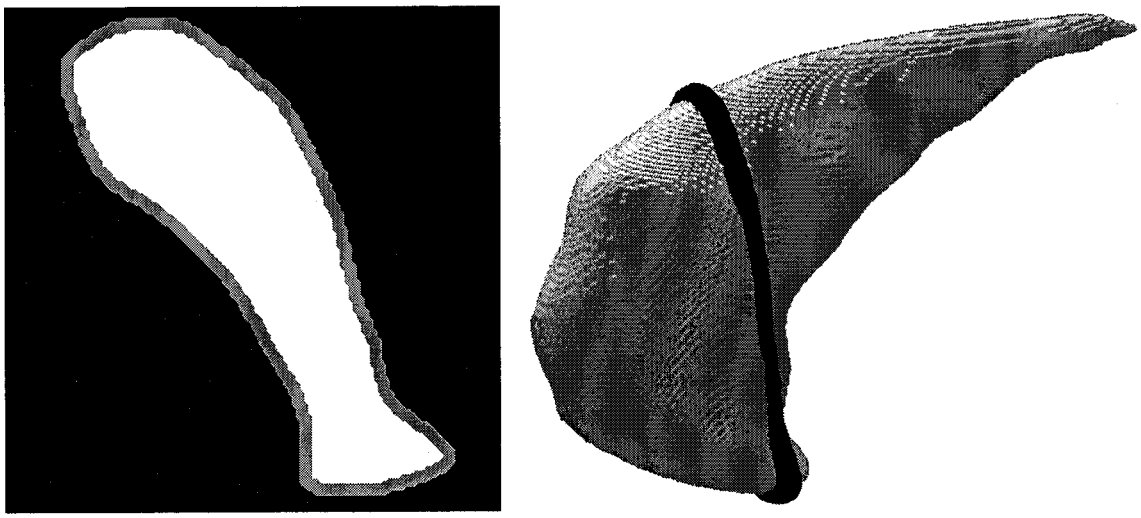
Figure 4. Livewire Interactive seeding. (a) Original image. (b) with a set seedpoint (circle), proposed contours from the seedpoint to the user's cursor is displayed in real time. (c) A proposed contour is selected (green), and the process in (b) repeats.

In Livewire, the user specifies sparse seedpoints along an object boundary and the object is segmented in this piecewise fashion. When the user specifies a seedpoint with the mouse, the 2D version of Dijkstra's algorithm using the above optimization costs is

applied, resulting in a 'cost map' that contains the minimal cost from the starting node (pixel) to all pixels on the image. Since the optimal paths from the starting pixel to all other pixels can be quickly computed, the optimal contour from the starting node to the mouse cursor location is displayed as overlay above the image. This contour is continuously updated as the cursor moves and appears to 'snap' against the object's boundaries and other high-gradient regions. With this real-time feedback, users can retain full control over how the contour looks without sacrificing efficiency to manually trace the contour. Figure 4 illustrates this process.

If the temporary contour looks satisfactory, the contour segment is locked by selecting the current pixel, and this process is repeated until the entire object is segmented. For clearly visible objects, only sparse seedpoints are required, thus gaining efficiency. However, for images of poorer quality, more shorter-spaced seedpoints are needed in these corrupt image areas in order to retain algorithm accuracy.

To improve the efficiency of Livewire's graph search step, numerous modifications such as LiveLane [34] and Livewire on-the-fly [35][36], which limit the algorithm's graph search space, were proposed. While improvements in technology now allow Livewire to operate in real-time without these modifications, automated Livewire methods that emulate user input over multiple image slices in a 3D volume still benefit from an abbreviated graph search implementation [37]. Figure 5 depicts how a 2D Livewire contour is represented in a 3D volume.



(a) (b)

Figure 5. 2D Livewire contour on a caudate nucleus mask. (a) 2D contour (gray) segmenting a slice of the caudate mask. (b) This same contour (black) is shown in the context of the entire object.

Chapter 2

3D Livewire for Arbitrarily Complex Topologies

This chapter first presents a review for existing 3D interactive segmentation techniques (Section 2.1). This is followed by the details of the extension from 2D Livewire to 3D (Section 2.2). This technique's performance is then validated on synthetic data, real medical volumes, and in robustness tests (Section 2.3).

2.1 Interactive 3D Segmentation

The incorporation of interaction in segmentation has been proposed for a large variety of methodology families. One such family that support or can be extended to 3D or user interaction include parametric, explicit [13][12][38][39][40][41][14], or implicit (e.g. level-set based) [15][16][17] energy minimizing deformable models. However, these models are prone to convergence to local minima. Active contours that converge to a

global minima have been developed [42][43]; however, these rely on a coarser discretization of the search space, succeeded by graph-theoretic optimization procedures that are less amenable to user interaction. User-interaction in level-set approaches [44] is not straight-forward because the contours in between the constraint points can be unpredictable and many task attempts may be needed to find an acceptable contour. Further, level-set approaches typically require more computations than other deformable models since contours on 2D images (1D manifolds) and surfaces in 3D images (2D manifolds) are represented using a higher dimensional (signed distance transform images), thus increasing the complexity of the problem. Also, employing graphics processing units (GPU's) to perform level-set calculations [45] may be needed to achieve interactivity [46], especially for 3D images. Other recent active contour methods that incorporate user intervention include [47][25]. In Yushkevich *et al*'s approach, implemented into the tool called ITK-SNAP [47], user-interaction is used for initializing an evolving 3D active contour, for setting up parameters, as well as for manual post-processing. However, while the initialization is largely graphical, user knowledge of this method's computational part is required for selecting cost parameters, and using imprecise initialization parameters can cause the contour to deviate. Furthermore, the user has no steering control during the curve evolution. In McInerney *et al*'s 'Sketch Initialized Snakes' [25], the user uses a graphical tablet to quickly initialize a 2D Snake, which is then automatically optimized. However, this approach requires specialized input hardware. Also, its computational efficiency, reproducibility, and tolerance to user error were not reported.

In addition, while interaction with 2D deformable models is straightforward, incorporating intuitive user-guidance and real-time visualization for 3D deformable surfaces or meshes remains a challenging human-computer interface problem. One approach to circumvent this problem is to iterate an interactive 2D algorithm on each slice in the 3D volume [48][49]. However, it is highly inefficient to constantly repeat this task, and without correlating computations within adjacent slices, segmentation results are often jagged (similar to manually delineated ones) when viewed from other orientations. By only requiring sparse user-guided contours or points to dramatically reduce interaction time, there exists many automatic surface reconstruction methods [50][51], some of which can even generate objects with arbitrary topologies [52][53][54]. However, their main drawback is that they do not consider image-based information; rather, they draw point connectivity and object topology conclusions based on the locations of the surface points alone. Also, their computation time is considerable and does not allow for user intervention during the task should mistakes occur. Similarly, Saboo *et al*'s 'GeoInterp' method uses sparsely interleaved manual segmentations in one orientation to initialize a geodesic Snake [55]. However, the user-input is not actually used as a hard constraint, and the Snake optimizes its shape without considering voxel intensities.

An alternative to deformable models is the 'Graph Cuts' approach, originally proposed by Boykov and Jolly [56] and further developed in [22]. Here, 'cuts', or globally optimal segmentations, are computed using manually specified foreground and background seeds (hard constraints) and boundary/region information (soft constraints). Refinement of the 'cut' can be made using additional user-placed seedpoints. This

method offers interaction simplicity especially in the 2D case; however, its main drawback is that because the seedpoints lie in the region bodies and not on the boundaries, results can be unpredictable along weak edges. Graph Cuts results will also vary depending on the soft constraint weighting and choice of seeds. Similarly, Rother *et al.* proposed the 'GrabCut' method that performs an iterative Graph Cut algorithm that decreases the amount of user interaction required [28]. Using the same interaction scheme, Grady proposed a method, random walkers, where all pixels in an image are assigned to each of the hard constraint seedpoints based on a probabilistic measure [29]. However, this method still shares the same limitations of the original 'Graph Cuts'. Also, 3D visualization during the segmentation task remains a challenge with this interaction scheme because user input is done in 2D but the segmentation task is not broken into 2D modules.

In extending Livewire to 3D, several methods requiring only sparse 2D contours were proposed, but they only consider image slices in one orientation. Souza *et al.* proposed a hybrid approach between Snakes and Livewire by projecting seedpoints from a previous adjacent slice onto the current slice and then refining their locations [57]. Similarly, Schenk *et al.* proposed an approach which takes sparsely spaced Livewire contours and interpolates and optimizes the contours in between using minimal cost paths [58]. Also, Malmberg *et al.* [59] proposed a method to bridge sparsely separated Livewire contours using haptic feedback and the image foresting transform [60]. However, special equipment such as a haptic device and a stereo-capable monitor is required. Moreover, image smoothness in orthogonal directions is not ensured, and medical images often contain objects with complex 3D shapes (e.g. deep concavities, protrusions, non-spherical

topologies, branching), which none of these parallel-slice approaches are able to effectively handle.

An approach was put forth by Falcao *et al.* to extend 2D Livewire to 3D by utilizing 2D contours on oblique slices to automatically mimic 2DLW on all slices in an orthogonal direction [61]. However, considerable user supervision and knowledge regarding the object's exact topological features are required to break a complex object down into 'slabs', which are groups of consecutive slices along the axis of automatic computation where the sub-object exhibits constant topology. The restrictions on these initial setup steps and on the selection of slices for 2DLW are critical to correctly segment each slab properly. Moreover, the intersection of these 2D contours with each slice in each slab generates seedpoints that need to be manually ordered in a clockwise or counter-clockwise fashion before they can be fed into the automated Livewire process. More recent 3D Livewire methods mitigate some of the above setup steps by using orthogonal 2D Livewire contours instead of oblique contours [62][63]. In Lu *et al.*'s 3D Livewire approach [62], seedpoint ordering is more automated than in [61], but it requires the projection of a manually-supplied reference contour onto adjacent slices. In Hamarneh *et al.*'s approach [63], seedpoint ordering is automatically computed using an algorithm based on turtle graphics [64] (part of Logo programming language) without additional image-based or user-supplied information. However, while both of these methods [62][63] do not require the complicated interaction steps of [61], these methods fail to address the problem of segmenting objects of arbitrary topology. Though these semi-automated methods presented reasonable solutions for certain segmentation tasks, their limitations highlight the need for a robust 3D segmentation approach that can

natively handle complex shapes of arbitrary topology, while at the same time still offering the advantages of user control, efficiency, accuracy, intuitive operation, and minimal user supervision.

2.2 Extending 2D Livewire to 3D

The user begins the segmentation process by performing sparsely separated 2D Livewire (Section 1.4) segmentations on slices in any two orthogonal orientations. These 2D contours are then used to determine the Livewire seedpoints to be used in the third orthogonal orientation by intersecting these contours and the unseen orthogonal slices (Section 2.2.1). These intersection points are pre-processed to increase robustness (Section 2.2.3) and are then used to create a 'turtle map' which consists of orthogonal line segments (Section 2.2.2). Our 'turtle' point ordering algorithm is then applied to this map such that the resulting ordered points mimic the sequence of points a user would select during a Livewire segmentation, but now in a fully automated manner on the unseen slices. Since these new seed points are a subset of the contours previously approved by the user, they are therefore a suitable choice of seedpoints for guiding the Livewire segmentation. In this scheme, user-generated contours that are circumscribed inside another contour are automatically flagged, and such flags are used to split and merge sections of the turtle map (Section 2.2.5). By doing so, multiple closed contours and objects with non-spherical topologies such as, for example, a vertebra (which has a toroidal topology due to the spinal canal) can be processed correctly. Figure 6 and Algorithm 1 summarize this approach.

Algorithm 1 Determining the segmentation for unvisited slice $S_{x,y_0,z}$ in xz given M input contours $\{C_{x_m,y,z}\}$ in yz and N input contours $\{C_{x,y,z_n}\}$ in xy. Note that it is a trivial change to process unseen slices in xy and yz.

Require: $\{C_{x_m,y,z}\}$ and $\{C_{x,y,z_n}\}$ on slices $S_{x_m,y,z}$ and S_{x,y,z_n}. x_m and z_n represent arbitrary slice indices of x and z respectively.

Ensure: Closed contours $\{C_{x,y_0,z}\}$ on slice $S_{x,y_0,z}$.

$TurtleMap \Leftarrow 0$. *Turtlemap* is the 2D area on which turtle tracks are placed.

$TurtleMap \Leftarrow Algorithm2(\{C_{x_m,y,z}\}, S_{x,y_0,z})$, building horizontal tracks of *TurtleMap*.

$TurtleMap \mathrel{+}= Algorithm2(\{C_{x,y,z_n}\}, S_{x,y_0,z})$, building vertical tracks of *TurtleMap*.

$P^a_{ord}(x,z) \Leftarrow TurtleMap(x,z)$, ordering seedpoints using Turtle algorithm (Section 2.2.2).

A = number of separate contours on $S_{x,y_0,z}$, where $A = \max(a)$.

for each $P^a_{ord} \in (1..A)$ **do**

$C^a_{x,y_0,z} \Leftarrow P^a_{ord}$, using abbreviated graph search (Section 2.2.4).

end for

$\{C_{x,y_0,z}\} \Leftarrow \{C^{a=1}_{x,y_0,z}, C^{a=2}_{x,y_0,z}, C^{a=3}_{x,y_0,z}, \ldots\}$, all contours with $y = y_0$ are found.

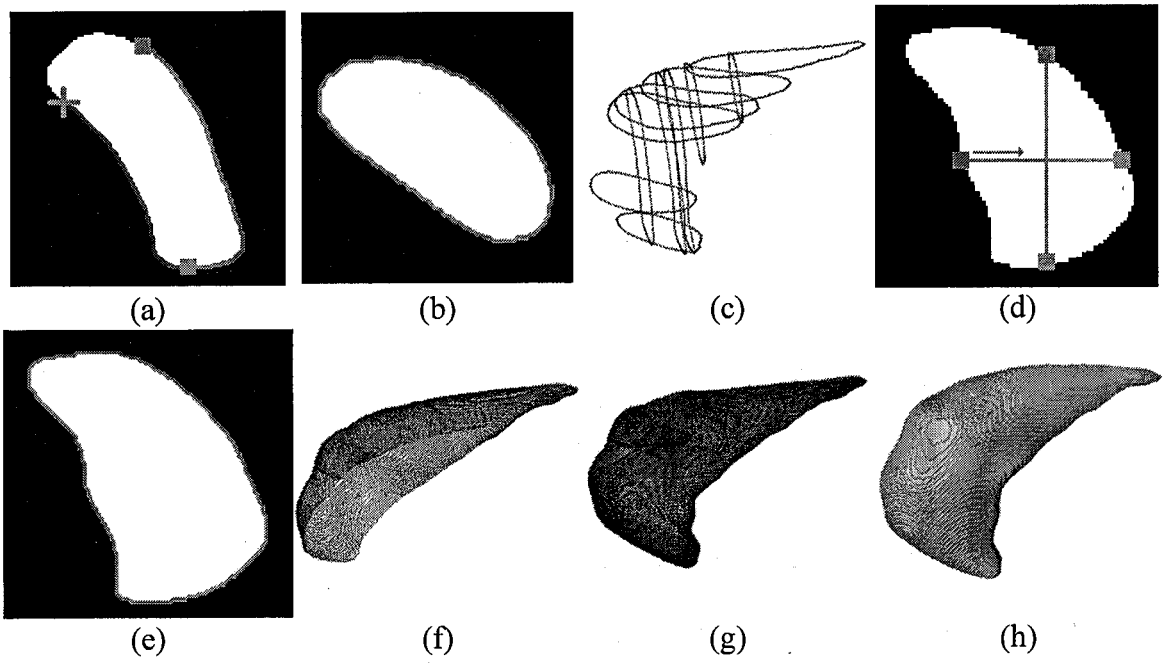


Figure 6. Overall steps of this algorithm, shown on a binary image for clarity. (a)-(b) Seedpoints (grey squares) are selected in user-guided Livewire contours on orthogonal slices. User-approved segments are in green, and red contours represent the proposed, 'live' segment during the Livewire task (crosshair denotes cursor location). (c) 3D plot of 11 user-guided contours. (d) Ordering automatically generated seedpoints (gray squares) on a slice in the third orthogonal orientation using our turtle-based algorithm (Section 2.2.2) results in the contour in (e). (f)-(g) 3D plots of automatically generated contours at mid-task and at task completion (125 contours), respectively. (h) Surface rendering of the segmentation result.

2.2.1 Automated Seeding

To perform an automatic Livewire segmentation, seedpoints need to be determined. These intersection points are simply the intersection between the 2D Livewire contours and the unvisited orthogonal slice that this automatic step is taking place on. For example, if 2D Livewire is used to create two contours C_{x_0yz} and C_{xyz_0} on arbitrary *yz* and *xy* slices respectively, then given a slice S_{xy_0z}, in the *xz* orientation at index y_0, the intersection points I_{xy_0z} between C_{x_0yz} and S_{xy_0z}, and J_{xy_0z} between C_{xyz_0} and S_{xy_0z} can be calculated as

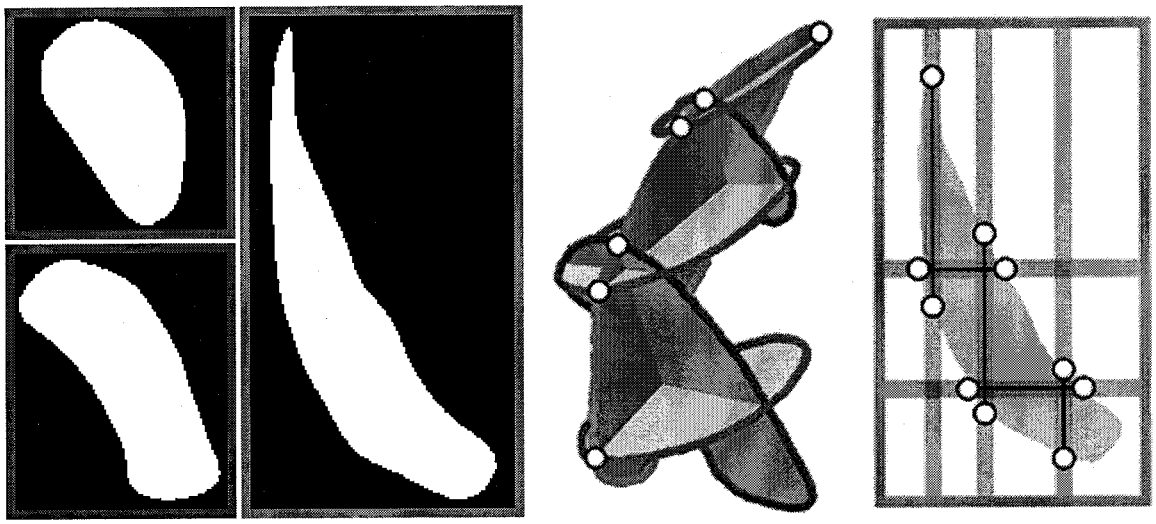
$$I_{x,y_0,z} \Leftarrow C_{x_0yz} \cap S_{x,y_0,z} \qquad J_{z,y_0,z} \Leftarrow C_{xyz_0} \cap S_{x,y_0,z} \ .$$

Similarly, the following equations define the intersection points I and J on slice S if different orientation combinations are chosen:

$$I_{x,y,z_0} \Leftarrow C_{xz} \cap S_{x,y,z_0} \qquad J_{z,y,z_0} \Leftarrow C_{yz} \cap S_{x,y,z_0}$$

$$I_{x_0,y,z} \Leftarrow C_{xy} \cap S_{x_0,y,z} \qquad J_{z_0,y,z} \Leftarrow C_{xz} \cap S_{x_0,y,z}$$

Figure 7 illustrates this determination mechanism.



(a) (b) (c)

Figure 7. Orthogonal contours in (a) (red and blue) intersect with a slice in the third orientation (green) in 10 different locations, as shown in (b). These intersections on the green slice in (c) become seedpoints on a turtle map (Section 2.2.2).

2.2.2 Automatic Seedpoint Ordering

In order to mimic a user-guided 2D Livewire segmentation task in an automatic fashion, the seedpoints, *I* and *J*, need to be ordered either clockwise or counter-clockwise in the 2D space of slice *S*. To accomplish this, seedpoints on the 2D space which belong to the same user-guided Livewire contour subset are paired and connected by lines (tracks). Since there are seedpoint contributions from two orthogonal orientations, these tracks, will themselves be orthogonal on slice *S*. Algorithm 1 summarizes this step and Figure 8 illustrates the final result, called a 'turtle map'.

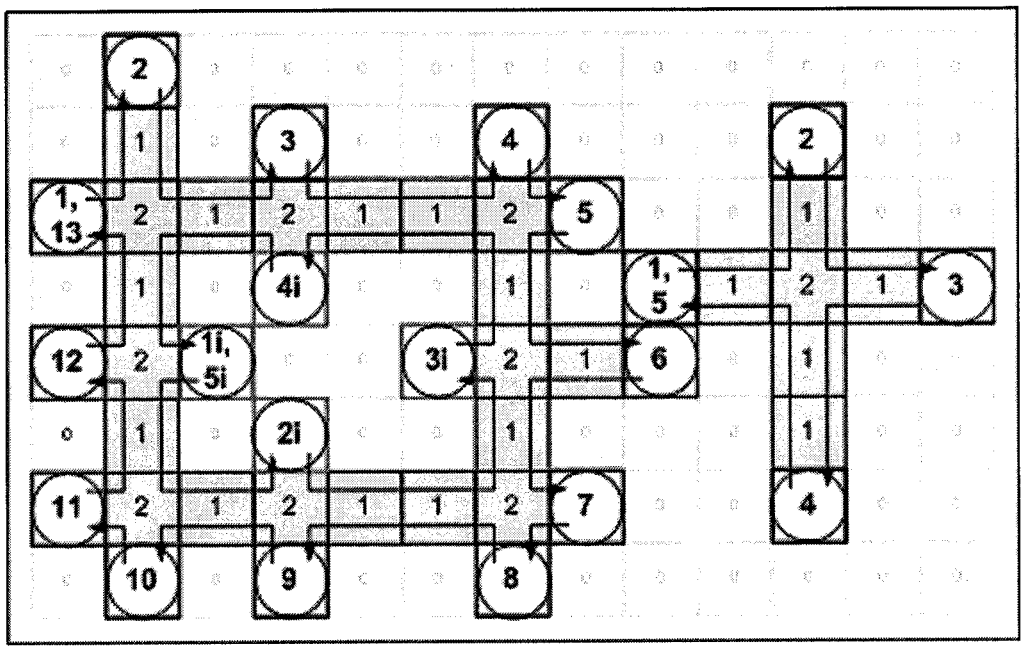


Figure 8. Example seedpoint map showing outer contour seedpoints (red), inner contour seedpoints (green with 'i' suffix), and a disjoint object's seedpoints (blue). For each contour, the turtle object starts at the first point and follows the tracks (in grey) according to its rules in Section 2.2.2, visiting other seedpoints in the order shown. Track values of '2' denote track intersections.

To traverse the turtle map and determine the correct ordering of seedpoints, an algorithm based on turtle graphics [64] is employed. Turtle graphics is based on the Logo programming language, and its main idea surrounds a directional turtle object that can only move forward and change directions on a graph-based system. Here, the turtle object begins at an arbitrary seedpoint and moves forward along the orthogonal tracks, turning to its left if it encounters a track intersection and reversing direction when it encounters another seedpoint. The sequence in which the seedpoints are visited determines their order (Figure 8). This process is repeated if there are multiple closed contours found on the same unseen slice, and this method keeps track of which seedpoints have been visited so that they are not encountered again.

While turtle map tracks usually intersect in a '+' like shape (Figure 9a), oftentimes turtle maps can exhibit 'T' junctions (Figure 9b) and 'L' junctions (Figure 9c),

as determined by how the user-guided contours intersect with this unseen slice. Here, our proposed turtle algorithm detects these situations and resolves them correctly by altering the turtle's movement rules and ensuring these seedpoints are not duplicated in the resulting seedpoint list.

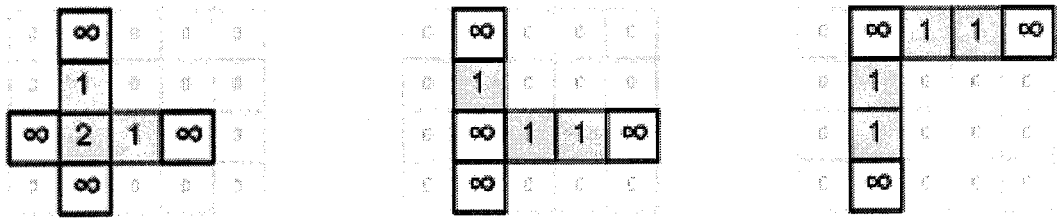
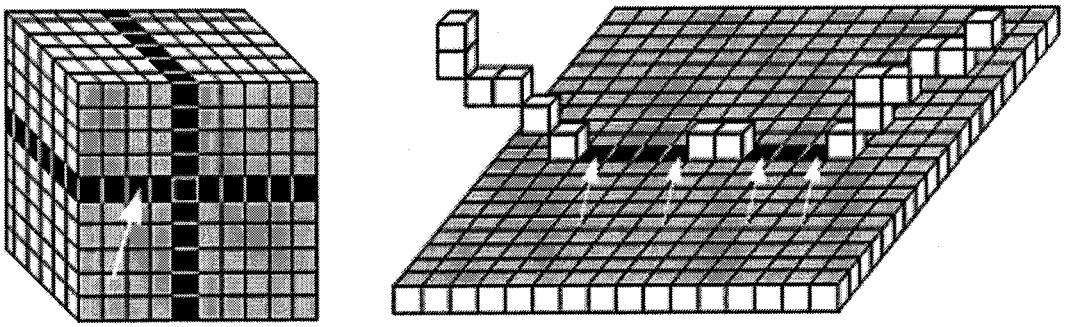


Figure 9. Possible turtle map intersections resulting from seedpoint locations, denoted by 1, include (a) the basic '+' junction, (b) the 'T' junction where a seedpoint overlaps a track, and (c) the 'L' junction where two seedpoints overlap. Pixels with values '1' and '2' represent non-seedpoint tracks and track intersections respectively.

2.2.3 Intersection Point Pruning

Ideally, a 2D contour would intersect with an unvisited orthogonal slice at an even number of locations, described by Hamarneh *et al.* as 'entering and exiting' the object [63]. However, objects with cusps can cause singular intersection points to exist as well. Also, while a user-guided contour will always be orthogonal to the slice in question, contiguous colinear contour pixels may intersect with this slice [62]. In the extreme example of a cube, a user-guided contour (a square) may be orthogonal to an unseen slice, but their intersection may comprise the entire square side of the contour (Figure 10a). To combat this, our proposed method assumes intersection points appear in cluster(s) or occupy consecutive pixel locations such as in Figure 10. Since the intersection points found between each user-guided Livewire contour and the orthogonal unvisited slice will always be a horizontal or vertical line on the slice, these colinear

points are first sorted in ascending pixel location order. Next, by traversing these points, cluster boundaries are easily found by determining the non-consecutive pixel location values. Knowing where clusters start and end allows us to prune the unnecessary points in between. An exception to the rule is when only one cluster is found, which corresponds to a cusp (singular point). In this case, the start and end of the cluster are kept and the middle points are discarded. With the extraneous contiguous points removed, the desired case of having an even number of intersection points is achieved. This allows for each automatically processed slice to be independent of all other parallel slices and to not require a reference frame [62]. Therefore, shape and topology changes (e.g. branches, cusps, saddle points) not observed in adjacent slices can now be seamlessly detected without further user supervision.



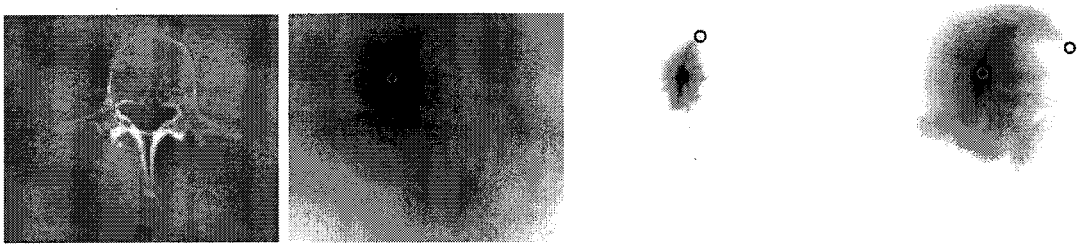
(a) (b)

Figure 10. Contiguous intersection points. (a) User-guided Livewire contours (black voxels) will intersect with the cube's end slice (gray) in multiple contiguous locations. (b) A contour intersecting with an orthogonal image slice (gray) creates two clusters of contiguous points (black). White arrows denote which pixels are kept after the pruning algorithm (Section 2.2.3).

2.2.4 Efficient Graph Search for Pre-determined Seedpoints

Since the seedpoints determined from orthogonal contours are pre-determined, no user-interaction is required and thus, an exhaustive 2D search using Dijkstra's algorithm

for each seedpoint is redundant. Our solution is similar to [35], as our modified graph search algorithm terminates after the next target point in the ordered list of seedpoints has been reached. The computational savings originate from the order in which the graph search propagation is done: the propagation algorithm selects the unprocessed pixel with the lowest accumulative cost to be analyzed next [26]. For example, when the graph search propagates from seedpoint q to point r, the accumulative cost of r is C_r. At this point, all arbitrary pixels p with accumulative cost $C_p < C_r$ would have been found already; thus the path from r back to q is guaranteed to be globally optimal. Figure 11 illustrates the impact of this technique. The graph search algorithm favours propagation along high-gradient edges and will largely ignore homogenous regions because seedpoints tend to be on or very close to gradient edges. Another advantage is that the computational savings now depend on the distance between seedpoints and not image resolution.



(a) (b) (c) (d)

Figure 11. Graph search required per pre-determined seedpoint. (a) Vertebra. (b) Full cost map needed per seedpoint (circled). Darker areas indicate lower cost. (c)-(d) Abbreviated graph search algorithm terminates when the next pre-determined seedpoint is reached.

2.2.5 Handling Arbitrary Topology

Anatomical structures often exhibit non-spherical topologies, concavities, and protrusions. For convex objects (e.g. sphere), it is guaranteed that there will only be one

or two clusters of seedpoints per input contour in unseen slices; thus, a turtle map can be easily generated using the technique described in our previously proposed framework [63]. However, for objects with concavities or protrusions (e.g. U-shaped tube), there may be situations where a slice captures multiple objects and its turtle map will show multiple disjoint groups accordingly (Figure 8). Since our current approach processes each group independently, multiple objects can be segmented concurrently, such as the left and right ventricles (Figure 14a). More complicated still are objects with non-spherical topology (e.g. torus), which none of the previous Livewire methods can handle. In order to correctly segment these objects, our method first identifies contours that are circumscribed within another, using pairwise comparisons on all user-guided contours of a given image slice. Let C_1 and C_2 represent two closed contours on the same slice. C_1 and C_2 are first converted to binary masks $M_{C_1}(x,y)$ and $M_{C_2}(x,y)$ respectively, where pixels inside the contour have a value of 1 and 0 otherwise. If

$$M_{C_1}(x,y) \cap M_{C_2}(x,y) = M_{C_1}(x,y) \quad ,$$

then C_1 is wholly situated inside C_2, and if

$$M_{C_1}(x,y) \cap M_{C_2}(x,y) = M_{C_2}(x,y) \quad ,$$

then C_2 is wholly situated inside C_1. This step is critical because these 'inner' contours delineate pixels that do not encompass the object of interest, but rather a hole in the object. Due to this, these contours and their derived seedpoints (Section 2.2.1) are flagged as 'negative', whereas the contours and seedpoints that actually delineate the object are flagged as 'positive'. Both 'positive' and 'negative' intersection points are used on the

turtle map; however, turtle tracks are only constructed between 'positive' seedpoints. In contrast, 'negative' seedpoints in effect negate a section of an otherwise longer track line, splitting the turtle map into two distinct parts. This process is illustrated in Figure 8, where seedpoints *2i* and *4i* negate the otherwise longer turtle track between seedpoints *3* and *9*. A central cavity results, which now correctly represents the toroidal object. This process is outlined in Algorithm 2.

Algorithm 2 Constructs the horizontal tracks in *TurtleMap* that is used for ordering the seedpoints found in Algorithm 1 for an unvisited slice $S_{x,y_0,z}$. For vertical tracks and if orthogonal unvisited slices $S_{x_0,y,z}$ or S_{x,y,z_0} are used, only trivial changes are required.

Require: An empty or incomplete *TurtleMap* array, G contours $C^g_{x_m,y,z}$ on slices $S_{x_m,y,z}$ respectively. x_m represents arbitrary slice indices of x.
Ensure: Construction of horizontal *TurtleMap* tracks.
for each $C^g_{x_m,y,z} \in (1...G)$ **do**
  $I_{x_m,y_0,z} \Leftarrow G^g_{x_m,y,z} \bigcap S_{x,y_0,z}$
  $Pairs^b(z_1,z_2) \Leftarrow I_{x_m,y_0,z}$, pruning algorithm in Section 2.2.3 to determine B point pairs
  for each $Pairs^b \in (1...B)$ **do**
    $z_1, z_2 \Leftarrow Pairs^b(z_1,z_2)$, extracting the z values from each pair
    $TurtleMap(x_m, z_1)$, $TurtleMap(x_m, z_2) = \infty$
    Let C be an arbitrary, closed, user-guided Livewire contour in the yz direction
    if $mask(C^g_{x_m,y,z}) \bigcap mask(C) = mask(C^g_{x_m,y,z})$ for any C then
      $TurtleMap(x_m, z_1 : z_2) = TurtleMap(x_m, z_1 : z_2) - 1$
    else
      $TurtleMap(x_m, z_1 : z_2) = \text{TurtleMap}(x_m, z_1 : z_2) + 1$
    end if
  end for
end for

2.2.6 Implementation Details

This proposed framework was developed in MATLAB (the MathWorks Inc., Natick, MA.) and offers the standard concurrent orthogonal views of a volume as shown

in Figure 12. As an overlay on top of the image data, user-guided contours are clearly demarcated in all views, regardless of their orientation. One criticism of this type of 3D Livewire extension [63] was that the slices used for user-guided contours had to be carefully selected otherwise the segmentation will fail [62]. By displaying these contours in this manner, our application gives users a clear idea of which areas have been segmented and which areas exhibit more topological features. In our findings, these feature-rich areas, if segmented correctly by the user, usually allow for higher accuracy. Also, this software feature is useful for visually judging the accuracy of the delineation result. Additionally, our user interface is able to display 3D plots of contours as well as a surface rendering of the object of interest after the 3D Livewire procedure is completed.

In our tool, additional features such as point deletion and automatic contour closing are available during the user-guided Livewire stage. Also, if the user selects a seedpoint erroneously, the segmentation process can be reverted to an earlier state, similar to the 'undo' command found in many common applications. While our technique is flexible and robust, errors are bound to occur due to human error and poor image quality. Our tool offers the undo operation described above, as well as the ability for users to remove entire automatically generated contours for re-computing. From the rendered result, users can quickly identify problematic areas, if any, and increase the segmentation accuracy by providing additional user-guided contours in these areas and re-running the 3D Livewire algorithm. For isolated refinement, users can also choose to overwrite the automatically generated contour(s) using the 2D Livewire procedure.

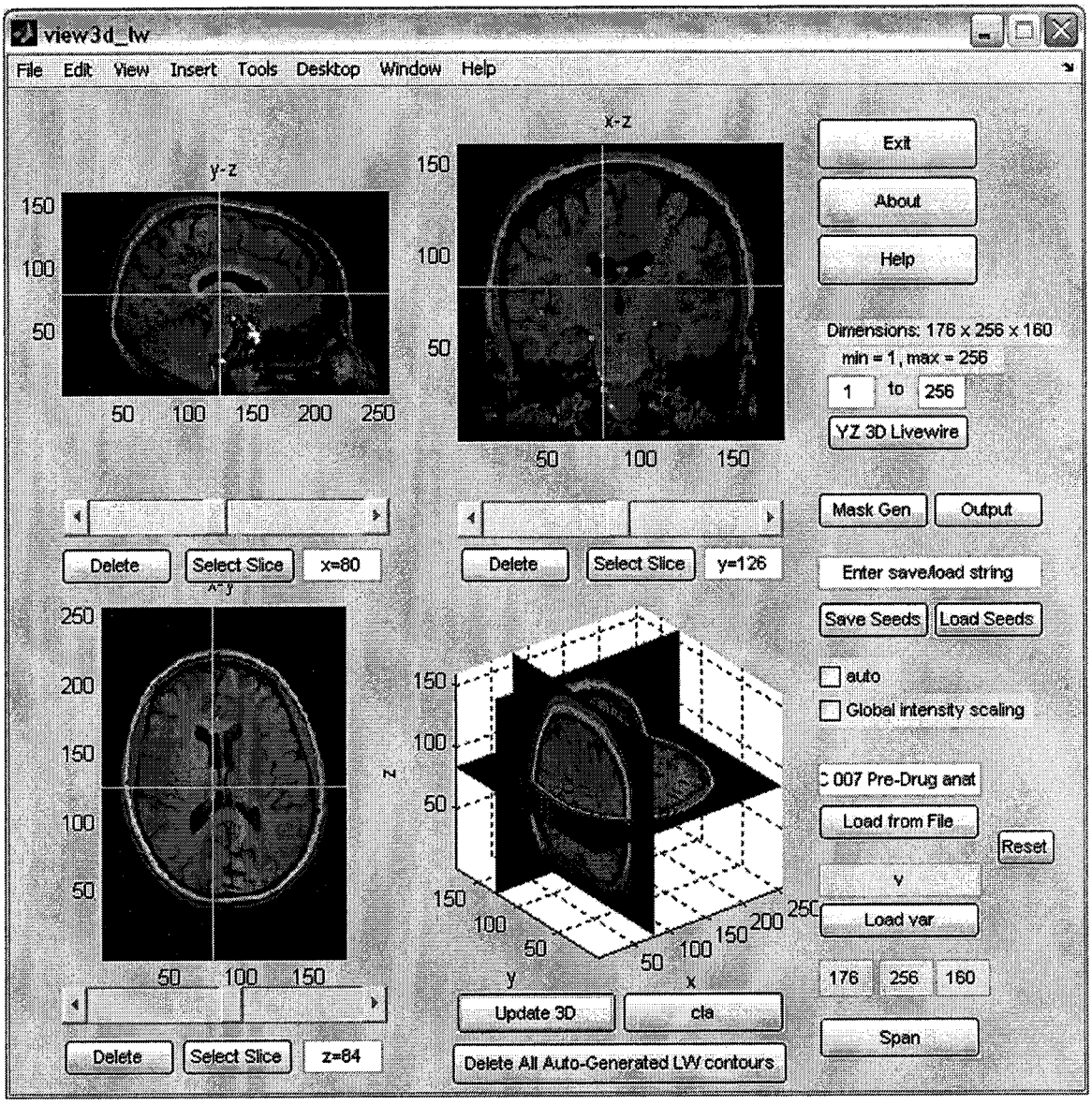


Figure 12. Screen-capture of the segmentation tool's graphical user interface during a segmentation task. Completed 2D contours are displayed in green for the three orthogonal views, providing feedback on segmentation accuracy throughout the segmentation task. Yellow lines indicate the current slice indices of the other two orientations.

2.3 Validation and Results

The proposed method was tested on both synthetic images (Section 2.3.1) and real medical image data (Section 2.3.2) to demonstrate its capabilities. The application's performance during these tasks was quantified based on the three main recommended

criteria for semi-automatic segmentation [9]. To report accuracy and reproducibility measurements, Dice similarity (voxel agreement) $C_{Dice} = 2vol_{sim} / (vol_A + vol_B)$ was used, where C_{Dice} is the Dice similarity coefficient. vol_{sim} is the sum of the voxels at the intersection between trial A and trial B, and vol_A and vol_B represent the sum of the voxels in trials A and B respectively. The Dice similarity coefficients were then averaged over all trials. Since our 3D Livewire method is deterministic and produces identical results given the same input contours, we measured reproducibility with different user-guided contours and seedpoints as input because not all operators will choose the same slices nor will they choose the same locations for seedpoints. The orientation of the 3D segmentation was kept constant, but operators with different familiarity of Livewire to were used. The untrained operators do not have prior understanding of topology and were trained for approximately 10 minutes on the segmentation tool and the objects they had to segment. Efficiency was calculated by comparing the time required for our technique to segment a 3D volume to the total time needed for performing 2D Livewire on each slice. Due to poor image quality or user mistakes, contour errors may occur with 3D Livewire; thus, the time it takes to correct such errors is included in the time measurements as well. Finally, algorithm robustness to increasing levels of additive white Gaussian noise (AWGN) as well as parameter sensitivity was also investigated in Section 2.3.3.

Synthetic data to validate our proposed method includes a mask of a left caudate nucleus (elongated object), a torus (toroidal topology object), and a fork-shaped object (branching object). We also demonstrate our method on real medical data such as the left and right ventricles from magnetic resonance imaging (T1-weighted MRI), a human

vertebra (computer tomography (CT) from the Visible Human Project (VHP) [65]), and both parts of the human pelvis (CT, also from VHP).

2.3.1 Synthetic Data Tests

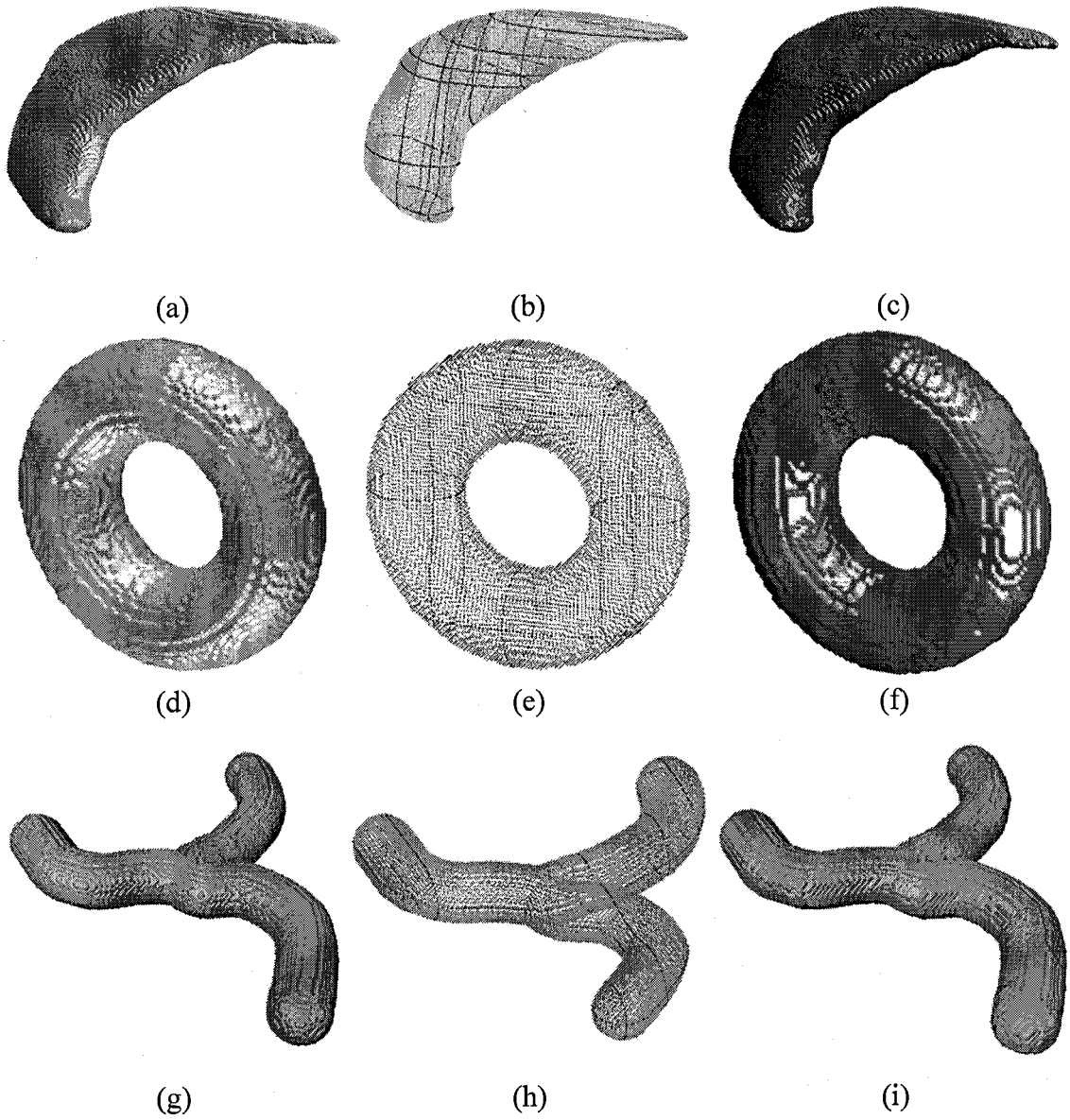


Figure 13. Results of our proposed segmentation method on synthetic data. (a),(d),(g) Rendering of a left caudate mask, torus, and fork object respectively. (b),(e),(h) 3D plot of user-guided contours (red) and automatically generated contours (light blue). (c),(f),(i) Surface renderings of the segmented synthetic examples above, using our proposed approach.

For the caudate nucleus (Figure 13(a-c)), very few user-guided contours are required to segment the body, and additional contours at the tail guarantees an accurate delineation. The torus example (Figure 13(d-f)) highlights our technique's ability to segment objects with non-spherical topology. For this scenario, only 8 user-guided Livewire contours are needed. If another orientation is chosen, only 6 user-guided contours would be needed. For the fork-shaped object (Figure 13(g-i)), only 5 user-guided Livewire contours were required to automatically generate 209 contours. The segmentation shows a smooth transition at the branching site. Table 1 summarizes our method's accuracy and reproducibility rates, averaged over multiple trials. Figure 13 shows the efficiency of our method for each computing phase. Our results show that our method is able to achieve these complex segmentation tasks in roughly 15% of the time it takes to delineate all slices using 2D Livewire.

We found that the reproducibility rates for these images are high because the input contours were computed using 2D Livewire, which has high reproducibility [26]. The minor differences between each trial largely depend on the accuracy of the chosen seedpoints, which varies by user. In terms of efficiency, total processing time naturally increases for volumes with high shape complexity. As the number of user-guided contours increases, so does the total amount of intersection points found on each unseen slice and ultimately, computation time. However, this higher processing time is counterbalanced by the fact that manual tracing of complex objects requires more user attention and segmentation time for an accurate delineation. We found that scaling a volume did not affect the number user-guided Livewire contours needed, as the same amount of these contours can still create valid turtle maps for all slices.

Table 1. Reproducibility and accuracy results of our proposed method, on both synthetic and real medical image data. Each averaged over 4 trials with the corresponding standard deviation. For the ventricles, vertebra, and pelvis examples, expert manual segmentations were not available.

Synthetic Data	Reproducibility %	Average Accuracy %
Left Caudate Mask	99.8 ± 0.2	98.7 ± 0.1
Torus	94.6 ± 2.7	95.3 ± 1.1
Fork Object	99.7 ± 0.1	97.1 ± 0.7
Real Medical Data	**Reproducibility %**	**Average Accuracy %**
L+R Ventricles	92.4 ± 3.1	
Vertebra	95.1 ± 2.4	
Pelvis (2 pieces)	98.1 ± 0.5	

Table 2. Task time reduction (s), achieved by our proposed method compared to performing 2DLW on each slice. Each step averaged over 4 trials. Standard deviation values between each set of trials are included. (I) User interaction time with our tool. (II) Automatic processing time of our tool. (III) Time required for manual corrections. (IV) Total task time of our tool. (V) Task time using 2DLW on all slices. (VI) Fraction of time (%) required for our tool compared to 2DLW on all slices (IV)/(V).

Synthetic Data	(I) User	(II) Auto	(III) Fix	(IV) Total	(V) 2DLW	(VI) Fraction (%)
Caudate	245.8 ± 87.5	86.8 ± 44.8	0	332.5 ± 128.5	2197.5 ± 713.7	14.9 ± 1.8
Torus	144.5 ± 56.6	18.3 ± 3.9	0	162.8 ± 59.2	778.8 ± 166	20.5 ± 3.5
Fork Object	146.5 ± 64.5	41.3 ± 3.4	0	187.8 ± 65.1	2310.5 ± 685.3	8.2 ± 1.3
Medical Data	**(I) User**	**(II) Auto**	**(III) Fix**	**(IV) Total**	**(V) 2DLW**	**(VI) Fraction (%)**
Left and Right Ventricles	428.3 ± 108.2	75.8 ± 14.4	0	504 ± 119.8	3431.8 ± 407.8	14.5 ± 1.8
Vertebra	684.5 ± 209.3	53.5 ± 5.9	162.5 ± 89.6	900.5 ± 221	3974.8 ± 599	22.5 ± 2.7
Pelvis (2 pieces)	1350.5 ± 512.3	467.3 ± 180.7	167.5 ± 66.3	1985.3 ± 653.4	6854.5 ± 1469.3	28.4 ± 5.3

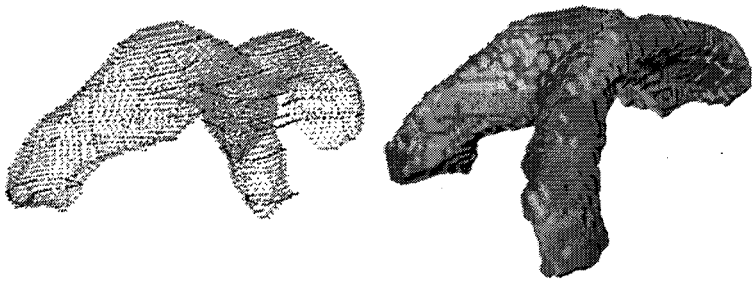
2.3.2 Real Medical Data

The first example presented here is a pair of ventricles segmented from an MRI volume (Figure 14(a-c)). Here, both disjoint structures were segmented during the same task, using a total of 24 input contours to automatically segment 105 slices (approximately 200 contours). In order to accurately capture the tail regions, a higher concentration of input contours was provided there. All object features were successfully captured, including the separation. Next, a vertebra was extracted from the male CT scan of the VHP [65] (Figure 14(d-f)). The human vertebra is toroidal, with pronounced protrusions. Also, the volume contains multiple vertebrae, and parts of two vertebrae often appear on the same slice. Here, our proposed method successfully extracted the vertebra using 17 input contours to segment 88 slices. Lastly, the human pelvis (Figure 14(g-i)), also from the CT scan of the VHP, was segmented using 80 input contours to segment 277 slices. For this example, due to the very thin bone characteristics at the ilium, the minimal number of input contours to create correct turtle maps at this area is difficult to achieve. This results in minor gaps in the segmentation, but these gaps were easily fixed with our tool using 2D Livewire to overwrite these problematic slices. The above examples were tested for reproducibility and efficiency over non-expert 2D Livewire done on all slices (Table 1 and Table 2). Similar to the synthetic data experiments, our proposed method is shown to be much more efficient than performing 2D Livewire on every slice in a real medical image and the segmentation is highly reproducible when provided different input contours.

Figure removed due to copyright permission.

MRI of brain from

A. Uthama, R. Abugharbieh, A. Travoulsee, and M. McKeown, Invariant SPHARM Shape Descriptors for Complex Geometry in MR Region of Interest Analysis, IEEE EMBS (2007) 1322-1325.

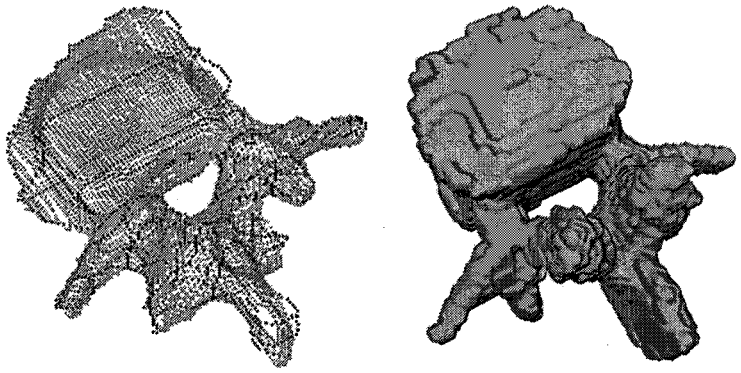


(a) (b) (c)

Figure removed due to copyright permission.

CT scan of vertebra from

M. Ackerman, "The visible human project," Proceedings of the IEEE 86, pp. 504–511, March 1998.

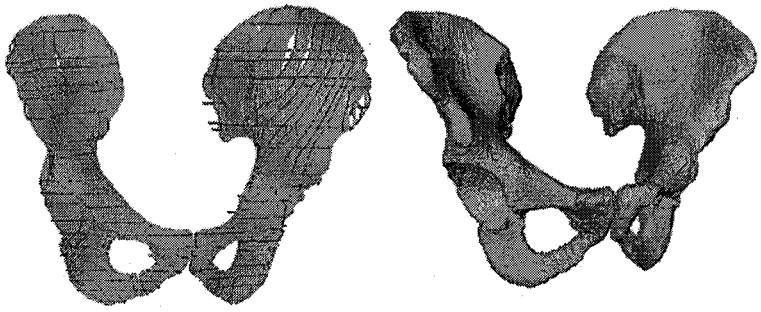


(d) (e) (f)

Figure removed due to copyright permission.

CT scan of pelvis from

M. Ackerman, "The visible human project," Proceedings of the IEEE 86, pp. 504–511, March 1998.



(g) (h) (i)

Figure 14. Results of our proposed segmentation method on real 3D medical data. (a),(d),(g) Original 3D images of a human brain (T1-MRI), spine (CT), and pelvic region (CT) respectively. (b),(e),(h) 3D plots of user-guided contours (red) and automatically generated contours (light blue). Twenty-four (red) used to segment 200 (cyan), 17 to segment 88, and 80 to segment 277 respectively. (c),(f),(i) Surface renderings of the segmented examples above, using our proposed method.

2.3.3 Analysis of Robustness and Parameter Sensitivity

Using consistent user-defined seedpoints for the user-guided contours, the caudate mask volume was subjected to incremental levels of AWGN and then segmented using

our proposed method. Accuracy levels were then plotted against the peak signal to noise ratio (PSNR) of the volume, defined as $PSNR = 20\log_{10} \max(ObjectIntensity)/\sigma_{noise}$. The segmentation results are shown in Figure 15a and accuracy results are shown in Figure 15b. As expected, the accuracy level decreases as the increasing amount of noise corrupts the image, but our method is able to recover much of the object even under high amounts of noise.

Parameter selection was not a problem in obtaining accurate results, and our implementation uses equal weighting for each term in our cost function ($w_{(1...5)} = 1$). Nevertheless, we investigated the effect of parameter sensitivity on the synthetic examples in Figure 13 to determine the change in accuracy when varying each weight value. We found that varying each weight by as much as ±50% did not change the accuracy by more than 3.1% in the test datasets.

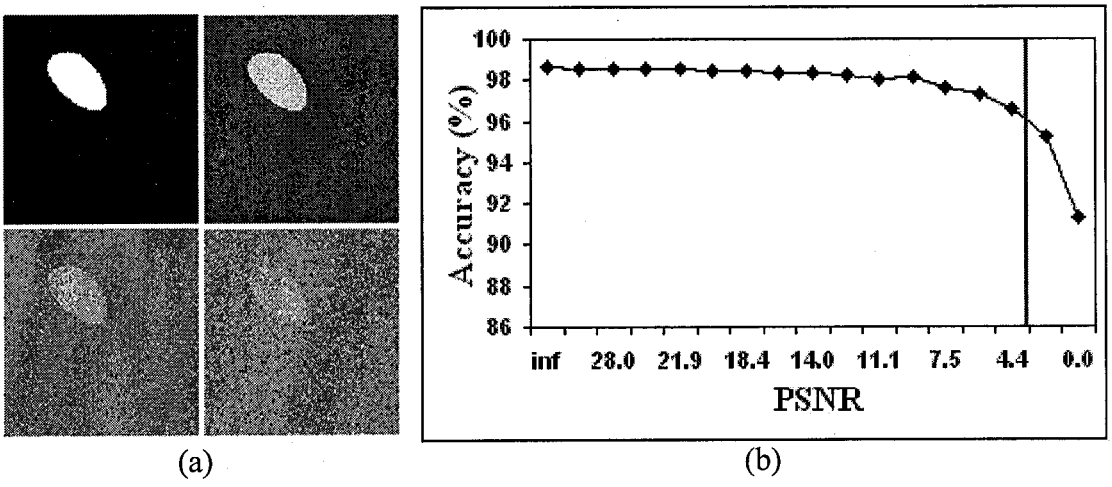


Figure 15. Our proposed method's performance reflected in segmentation accuracy as AWGN is progressively added. (a) Slices of a left caudate mask with increasing noise and PSNR levels of 1 dB,20.0 dB,6.0 dB, and 0 dB. (b) Accuracy level stays consistently high until very high noise levels occur, obscuring the ability of the user to choose reliable Livewire seedpoints.

Chapter 3

Live-Vessel: Extending Livewire to Vascular Data

This chapter first presents a review for existing vessel segmentation techniques (Section 3.1). This is followed by the details of the extension from 2D Livewire to 3D (Section 3.2 and 3.3). This technique's performance is then validated on synthetic data, real medical volumes, and in robustness tests (Section 3.4).

3.1 Vessel Segmentation Overview

In addition to the aforementioned difficulties in medical image segmentation, further complications in vascular images include vessel bifurcations and noise corruption of vessel edges and centres, disrupting an otherwise boxcar-shaped vessel profile of an ideal vessel. Noise can also create what appear to be gaps in vessels [66] that only experts familiar with anatomy can discern. In vessel segmentation, topological conformity

ensuring vessel tree connectivity is important for further analysis, such as discovering branching patterns or branch lengths and tortuosity [3]. Many such analyses require the determination of the vessel medial path (as opposed to vessel boundaries only). Vessel medial axis extraction facilitates other forms of shape analysis, data registration [7], and vessel boundary extraction itself [67][68].

Manual tracing by experts for vessel segmentation is considered accurate. However, it is extremely time consuming due complex vascular networks. Also, long segmentation tasks, particularly needed in vascular images, are commonly affected by user fatigue [9]. Therefore, some level of automation is necessary in this application.

Numerous methods have been proposed for vessel segmentation. Kirbas and Quek [69] provided a survey of several existing vessel segmentation methods. Automated vessel segmentation methods are diverse. One family of methods uses pattern recognition techniques combined with post processing steps such as pruning or pixel classification after preliminary vessel detection. For example, Ricci and Perfetti [70] used a line detector with a rotational interval of 30° which was coupled with a support vector machine (SVM). Similarly, Soares *et al.*'s filtering method [71] used Gabor wavelets at rotation intervals of 10° along with a Bayesian classifier. Staal *et al.* [72] employed ridge analysis [73] and a *k*NN classifier. Such classifier-based methods, however, require training which complicates their applicability to clinical use. Lam and Yan [74] used normalized gradient vector fields to locate vessel centerlines, and a Laplacian-based vessel detector was used to prune the result. However, gradient vector fields are highly susceptible to noise, especially when detecting thin vessels. A bigger limitation is that

none of the above methods enforces vessel connectivity constraints. This often results in broken vessels and the inability to determine accurate medials and vessel trees.

Another family of automatic vessel segmentation methods is based on energy-minimizing evolution of deformable models [75][76][77][78][79][80]. Implicit, level-sets based approaches (e.g. [75][76]), as well as topologically adaptable explicit models, such as T-snakes [38] or T-surfaces [81], are able to handle the complex vascular topology. However, initializing topologically adaptable models with a single seed may not segment distant branches, and using multiple distributed seeds does not ensure that all parts will merge to produce a correct vessel tree connectivity (having correct topology). Other methods that take advantage of vessel-specific properties include maximizing the gradient vector field flux to optimize the contour or surface [82] and modeling of the capillary force to segment small vessels [83]. In general, results of energy-minimizing approaches can be unpredictable and many trials and modifications of the model initialization and parameters may be needed. Also, these methods are not guaranteed to give a globally optimal solution. Globally optimal energy minimizing segmentation approaches have been proposed [84][42]. However, such approaches still suffer from restrictions: to the best of our knowledge none were specifically designed for segmenting tubular branching structures (or vessels); they simplify the cost terms to achieve global minima of convex functionals; or they discretize and limit the search space of possible solutions. The Graph Cuts [22] approach achieves a globally optimal segmentation (given a set of foreground and background seeds). Hraiech *et al.* [85] segmented abdominal aortic aneurysms using Graph Cuts. However their work is a direct application of Graph cuts without any vessel-specific extensions. Li *et al.* [86] used Graph Cuts to extract

optimal surfaces in tubular 3D objects. However, a volume has to be partitioned into carefully chosen regions in which the boundary surfaces can be unfolded in terrain-like surfaces. In general, Graph-Cut contours are may also be unpredictable since user interaction is obtained via interior region seeds and boundary constraints, so they typically require user refinement in the form of additional seedpoints, thus this method's ability to segment complex vascular networks, not just major tubular structures, is not yet proven.

Other approaches for automated vessel segmentation employ multiscale detection of curvilinear structures, which is effective in discerning both large and small vessels. In [87][88][89], the eigenvector associated with the minimum eigenvalue of the Hessian matrix across all scales was used to estimate the vessel direction at a given pixel as the direction with the smallest image intensity curvature. Aylward and Bullit [90] applied this directional information to traverse image ridges and estimate vessel centrelines. Building on [88][89], Frangi *et al.* [91] developed a "vesselness" filter using the Hessian matrix eigenvalues. Other approaches combined vesselness with vessel enhancing diffusion [92][93], region growing [94], cylindrical vessel models [95], and matched filtering [96]. Sofka *et al.* [97] later developed another vesselness measure using matched filters instead of Hessian-based techniques. Multiscale vessel detection proved useful in detecting vessels; however, to the best of our knowledge, only the maximal response across the scales was used at any particular pixel thus lacking regularization of scale along the vessel. An exception is the Vessel Crawlers approach [98], where an estimate of the scale is derived from the radius of the leading front of the crawler thus achieving a form of scale regularization, but without any global scale optimality condition. Placing

seedpoints along a vessel's medial axis can enable the implementation of vessel extraction methods based on path optimization techniques in the image domain. Finding minimal paths in 2D images was explored in [43]. Deschamps and Cohen [99] extended the approach to 3D, where a fast marching front propagation was implemented to approximate the medial axes of tubular structures in spatial dimensions (x,y,z). However, that work did not address the problem of identifying the boundaries of tubular structures. Young *et al.* [100] employed the vesselness filter of [91] with a front-propagation based algorithm to segment the vessel and extract its medial axis. However, the scale that achieves the maximum vesselness response for each pixel independently was used, and not the scale that minimizes the total cost of the path. A key point here is that the vesselness scale at each pixel should be chosen such that a globally optimal path is found, and not chosen to maximize the vesselness response at that pixel. Our proposed approach addresses this exact issue. The two vessel tracking methods that relate most to our proposed method are the following. Li and Yezzi [101] optimized for both spatial variables and a 'radius' variable, thus simultaneously determining the globally optimal medial and vessel boundaries. However, this method did not incorporate features of vesselness magnitude nor did it take advantage of vessel direction. Further, Wink *et al.* [27] applied Dijkstra's algorithm [31] to find the globally optimal medial and corresponding scale values. They used multiscale vesselness filtering that did not discard non-maximal response. The cost of selecting a certain scale at a particular pixel was inversely proportional to the vesselness response for that given scale at that pixel. However, this method does not take advantage of the vessel direction information and does not utilize edge information at a scale-dependant distance from the medial.

The Livewire segmentation framework [26] is an optimal path finding algorithm for image segmentation that emphasizes user involvement during the segmentation process. Such an interactive approach, unlike fully automated approaches, can be beneficial to vessel segmentation as it combines the advantages of intuitive user validation along with efficient computational speeds [9][18].

3.2 Livewire to Live-Vessel: Extending Graph Search to (x,y,r) Space

In our proposed method, rather than delineating a vessel by guiding a Livewire contour along the vessel boundary, which is inefficient, our technique enables the user to steer a contour along the centerline (or medial axis) of a 2D vessel. The contour along the medial the vessel is computed as the optimal path between two points in 3D space (x,y,r), where (x,y) are the image spatial coordinates at each medical node and r represents the corresponding radius values at those nodes. This approach reduces the amount of seedpoints needed, since a single optimal path defines three contours (the medial axis and two boundary contours on either side). In 2D Livewire implementations, given an image $I(p)$, where $p = (x, y)$, the only local path choices from node p are to one of the eight neighbouring pixels, $q = (x', y')$. In Live-Vessel, we optimize the medial axis path with respect to three variables: the two spatial variables x and y, and the vessel radius variable r. This extends the traditional Livewire graph search from 2D to 3D. However, since the medial axis in reality is in 2D space and cannot connect $p = (x, y, r)$ to $q = (x, y, r')$ where $r \neq r'$ (i.e. only a single radius value can be associated with each medial node), our 3D graph search is restricted accordingly (Figure 16). To accommodate for vessels that

dilate and constrict rapidly, the radius value can change to any other set of values (albeit penalized differently, as described in Section 3.3.4). This increases the computational complexity, but the optimal medial path and optimal vessel thickness are then guaranteed.

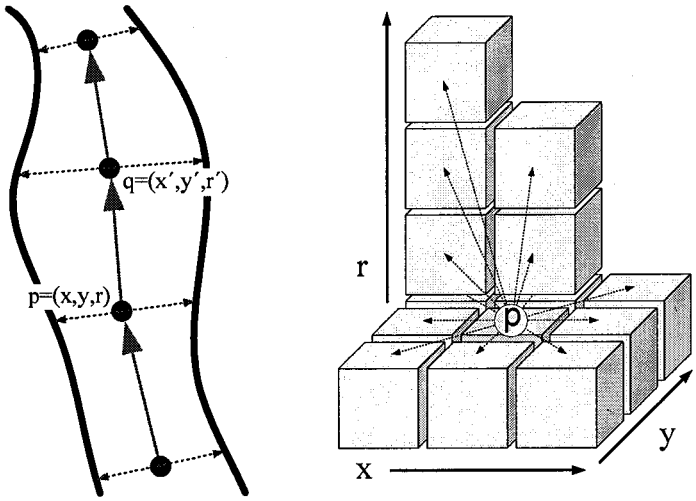


Figure 16. Live-Vessel's 3D graph search algorithm depicted in 2D (x,y) and 3D (x,y,r). (a) Medial path sequence (green arrows) with two neighbouring nodes $p=(x,y,r)$ and $q=(x',y',r')$, projected on the (x,y) plane. Red arrows denote the radius (r) dimension. (b) Alternative directions from p to q. Note that the next node on the path after $p=(x,y,r)$ cannot be $q=(x',y',r')$ if $x=x'$ and $y=y'$.

While traditional Livewire displays the proposed contour between two points, Live-Vessel displays the proposed medial path as well as its associated boundary contours. In addition to specifying physical coordinates (x,y) with the cursor at seed points, the radius (r) is increased/decreased via keyboard arrow keys. The user can then quickly determine the next seedpoint such that the current vessel segment is delineated correctly. One of Livewire's advantages, which Live-Vessel maintains, is the intuitive control of desired segmentation accuracy and efficiency given varying degrees of noise and object complexity. To actually extract the vessel, the medial path is first projected back onto the (x,y) plane. Each medial node has an optimal radius value r, and its preceding and succeeding nodes form a direction vector. These elements are used to determine the two vessel boundary points on either side of the medial node. Repeating

this for all medial nodes, all the boundary points of the vessel are found in sequence and converted into a segmentation mask. This process is illustrated in Figure 17 and Figure 18.

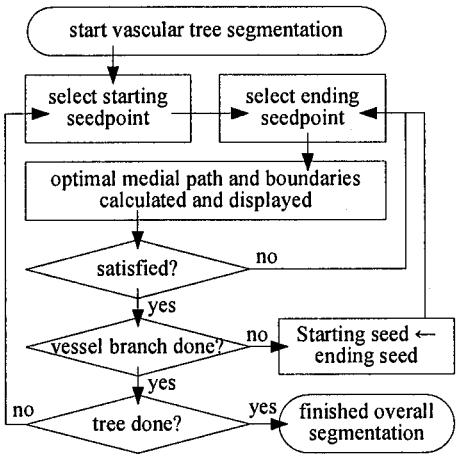


Figure 17. Flowchart depicting program operation from user point of view.

Figure 18. Overview of Live-Vessel operation. (a) Original Image. (b) Vessel boundary points (yellow) from a seedpoint (green cross) to the next potential seedpoint (red square) are graphically shown to the user for approval. (c) Segmentation mask is created from boundary points determined in (b).

3.3 Live-Vessel External and Internal Costs

We define the incremental cost function, which describes the cost from node $p = (x, y, r)$ to a neighbouring node $q = (x', y', r')$, as:

$$Cost(q, p) = w_1 C_V(p) + w_2 C_{Ev}(q, p) + w_3 C_{Ie}(p) + w_4 C_R(q, p) + w_5 C_S(q, p)$$

C_V is associated with Frangi *et al.*'s multiscale vesselness filter [91]; however, quite importantly, our vesselness response at p is evaluated at different scales rather than choosing the scale with the maximal response at (x,y) (see Section 3.3.1). C_{Ev} refers to the cost of vessel direction change between p and q, and C_{Ie} is a measure of the medial node fitness assessed using the image edge evidence at a scale-dependent distance. Two smoothness cost terms (C_R and C_S) are used to penalize paths where the radius, r, or spatial variables, (x,y), fluctuate rapidly (i.e. the two terms regularize the vessel thickness and medial axis, respectively). Note that evaluating C_v, C_{Ev}, C_{Ie}, and C_R depends on scale, which is a variable being optimized for during our graph search, whereas C_S does not depend on scale. Each cost term is normalized to lie in the range [0,1] and is weighted by $w_i, i \in \{1..5\}$. All cost terms are explained in Sections 3.3.1-3.3.4 and summarized in Figure 19.

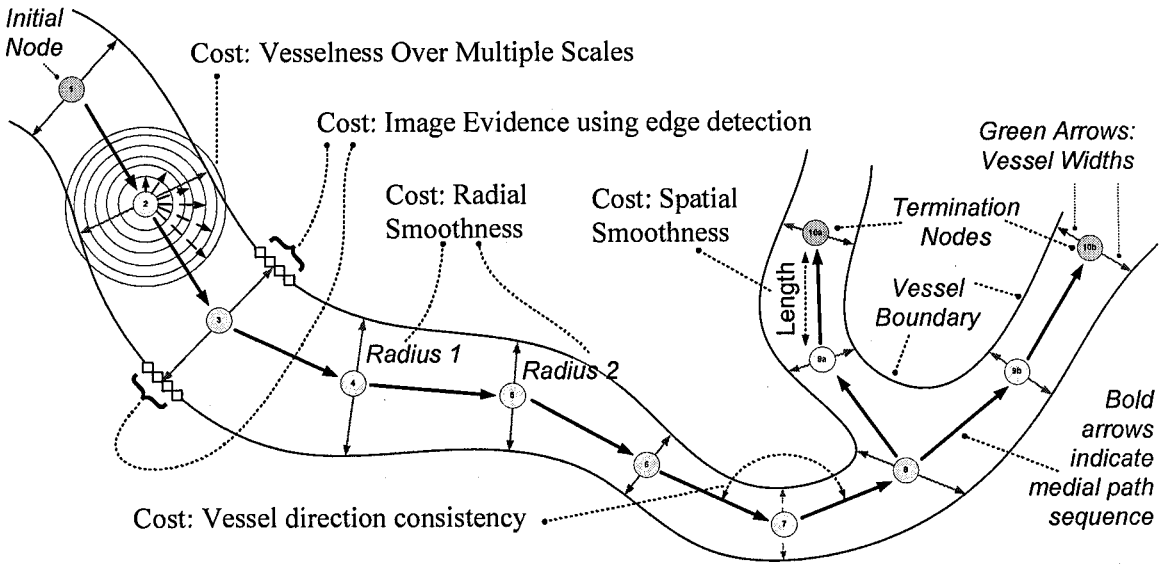


Figure 19. Graphical representation of the cost terms in Live-Vessel's minimum path search.

3.3.1 Vesselness Cost

To detect curvilinear structures in 2D at a given scale σ, the image is first convolved with a Gaussian kernel with variance σ^2. Then, the ordered eigenvalues $|\lambda_1| \leq |\lambda_2|$ of the 2 × 2 Hessian matrix H_σ for each pixel can be used to determine whether a pixel lies on a vessel of that scale [88]. Table 3 summarizes the eigenvalue conditions for vessel detection at a given pixel.

Table 3. Needed eigenvalue conditions from the Hessian matrix at a given pixel for vessel detection.

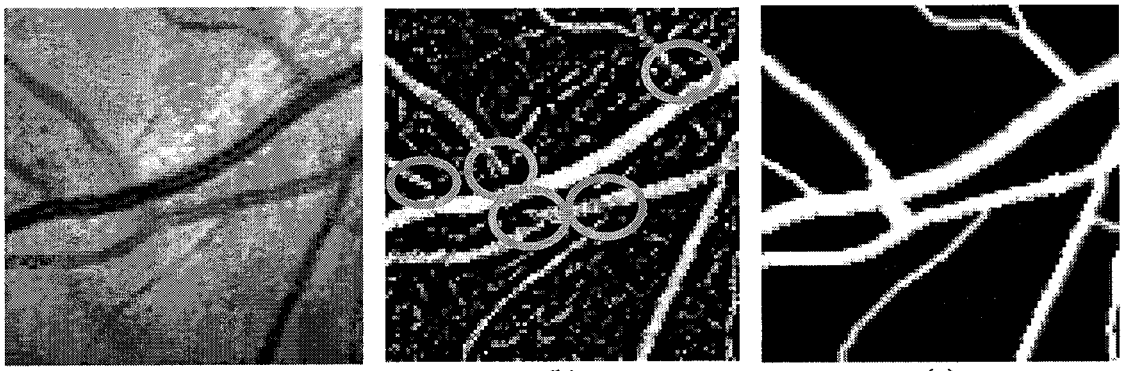
Eigenvalue condition	Visual interpretation
$\lambda_1 < 0$, $\lvert\lambda_1 / \lambda_2\rvert$ = large	Dark background, bright vessel
$\lambda_1 > 0$, $\lvert\lambda_1 / \lambda_2\rvert$ = large	Light background, dark vessel

In our implementation, we used the filter proposed by Frangi *et al.* [91] to quantify the likelihood of a dark vessel pixel into a 'vesselness' image feature. This filter is adopted as a cost term in Live-Vessel's optimization process as follows:

$$C_V(q) = V(\lambda_1, \lambda_2) = \begin{cases} 1 & if \lambda_2 > 0 \\ 1 - \exp\left(-\frac{R_\beta^{\,2}}{2\beta^2}\right)\left(1 - \exp\left(-\frac{T^2}{2c^2}\right)\right) & otherwise \end{cases}.$$

$R_\beta = \lambda_1 / \lambda_2$ represents the eccentricity of a second order ellipse and $T = \sqrt{\lambda_1^2 + \lambda_2^2}$. β and c affect filter sensitivity and have values 0.5 and 0.3 respectively. These values are similar to those used in other vessel filter studies [91][27].

Previous techniques proposed for detecting curvilinear structures using the Hessian matrix, except Wink *et al.* [27], chose the highest filter response for each pixel; however, this is susceptible to noise and does not enforce spatial continuity (Figure 20b). Our proposed method, in contrast, stores the results over a range of scales[1], separately, and optimizes for the scale-dependent variable r too. By combining the results with our other cost terms, we place restrictions on the relationship of neighbouring nodes; hence, ensuring robustness to poor image quality and bifurcations (Figure 20c).



(a) (b) (c)

Figure 20. Vessel bifurcation and noise poses problems for a maximal response vesselness filter. (a) Original image, magnified for clarity. (b) Maximal response vesselness filter output. Note the filter problems caused by vessel bifurcation or noise. (c) Segmentation result with our proposed method.

3.3.2 Vessel Direction Cost

The eigenvector $Ev(x,y,r)$ of H_σ, corresponding to λ_1, points in the direction of the minimum principal curvature, which estimates the vessel direction [87]. By choosing paths that minimize the change in direction of Ev, we can mitigate local noise and favour

[1] Defining boundaries as zero crossings of the Laplacian of a Gaussian intensity profile across the vessel yields $\sigma = r$, where r = vessel radius. This is derived by equating the second derivative of the Gaussian kernel to zero.

vessel directions that change smoothly. We therefore incorporate the cost term $C_{Ev}(q,p)$ from the incremental cost function and define it as:

$$C_{Ev}(q,p) = \frac{2}{\pi}\arccos\left|\frac{Ev(p)\cdot Ev(q)}{\|Ev(p)\|Ev(q)\|}\right| \quad ,$$

which evaluates the direction change from neighbouring nodes $q=(x,y,r)$ to $p=(x',y',r')$. From the vesselness filter, $Ev(q)$ and $Ev(p)$ (Figure 21) point arbitrarily in either directions of a bidirectional vessel (i.e. $\pm Ev(q)$ and $\pm Ev(p)$). The equation above gives the same cost regardless whether the proceeds with segmentation in either vessel directions: the medial path cost from a q to p node transition is equal to a p *to* q sequence. If an operand of a dot product is inversed, the dot product becomes negative but its magnitude is unchanged ($Ev(q)\cdot Ev(p) = -(Ev(q)\cdot -Ev(p))$); thus we use the absolute value operation above to cancel any potential negative signs and obtain the smaller angle between the vectors (Figure 21).

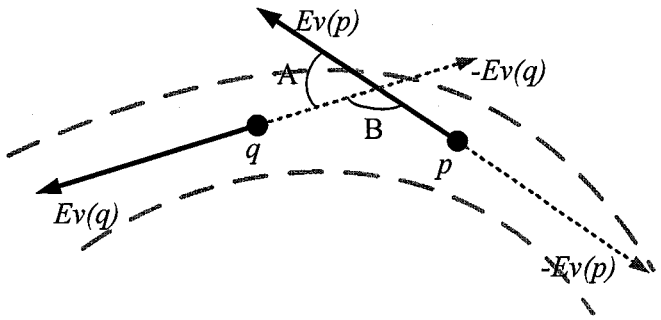


Figure 21. Eigenvector consistency cost $C_{Ev}(p,q)$ going from node $p(x,y,r)$ to $q=(x',y',r')$ is calculated using angle A, not angle B. Angle A is the smallest angle between $\pm Ev(p)$ and $\pm Ev(q)$.

3.3.3 Image Evidence Cost

Our proposed Live-Vessel also uses image evidence in the form of edge information to favour medial nodes that are located at the centre of a vessel cross-section of radius r. For this, we employ Canny, Laplacian of Gaussian (LoG), and gradient magnitude based edge detection and average their responses into $R(x,y)$. We chose these filters because Canny and LoG filters are less sensitive to noise, whereas the gradient magnitude filter does not involve pre-smoothing and thus complements the Canny and LoG filters by detecting weak structural edges.

Specifically, for each $p=(x,y,r)$ node in the image, we combine the vesselness direction $Ev(x,y,r)$ and $R(x,y)$ to define another measure of node 'medialness'. By finding the two unit vectors that are normal to Ev in the (x,y) plane and scaling them by r, we can determine two locations at which the vessel wall should be located. We thus retrieve the corresponding $R(x,y)$ at these points and their adjacent points $P_R=\{(x_i,y_i);i=1,2...N\}$ (N is the total number of points considered on both sides) in parallel directions to Ev (Figure 22a). The image evidence cost $C_{Ie}(p)$ in the incremental cost function is then defined as $C_{Ie}(p)=1-(1/n)\sum_{i=1}^{N}R(P_R)$ to average a potentially noisy response. This cost term is minimized for a medial node that has equidistant (at distance $=r$) vessel boundaries, which is expected for vessel medial nodes. Figure 22c-d show this cost's effect when other cost terms are not considered.

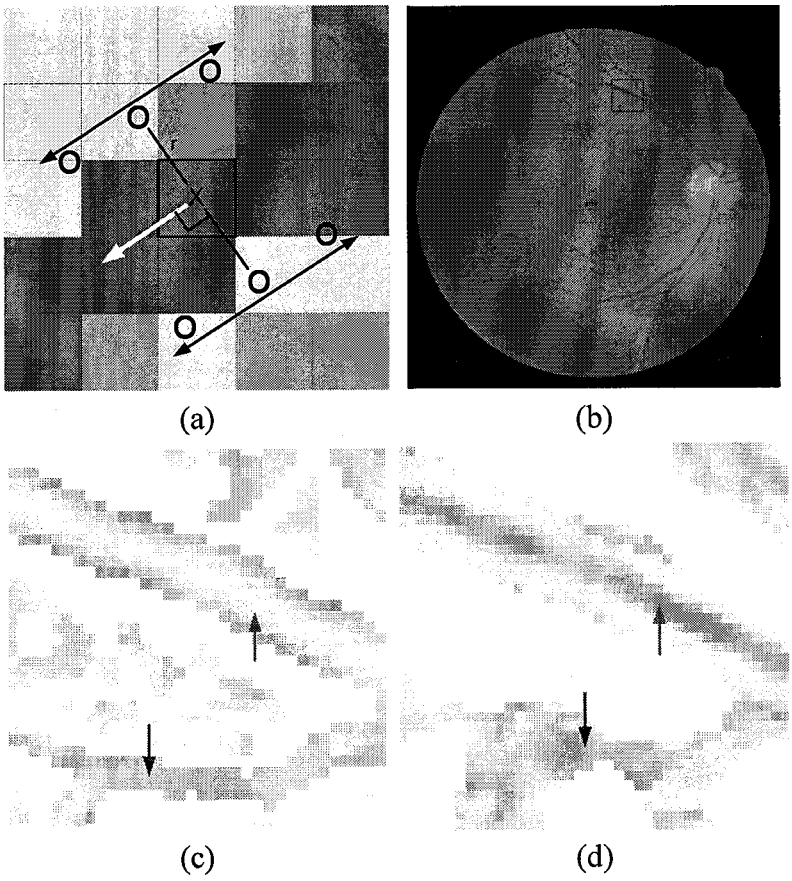


Figure 22. Image evidence cost and its effect. (a) points (O's) parallel to vessel direction (white arrow) at a scale-dependent distance *r* are tested for edge detection response. (b) Original retinal image. Red square denotes close-up area in (c) and (d). (c) Image evidence cost function $C_{Ie}(p)$ at a small value of *r* (darker means less cost). Medial nodes of thin vessels are prominent and medial nodes of large vessels exhibit higher cost. (d) Image evidence cost function $C_{Ie}(p)$ at a larger *r*. Medial axes of larger vessels become prominent and medials of smaller vessels exhibit higher cost and are dispersed.

3.3.4 Spatial and Radius Smoothness Costs

By encouraging the medial axis to be shorter, medial jaggedness is avoided and spatial smoothness is improved (Figure 23a). To this end, Live-Vessel imposes a cost proportional to the length of the medial axis. An incremental cost for each additional node added to the path (connecting points $q = (x, y, r)$ to $p = (x', y', r')$) is used, which accumulates during the graph search operation. This cost, $C_S(q,p)$ in the incremental cost function, is proportional to $\sqrt{(x - x')^2 + (y - y')^2}$.

Similarly, we penalize medial paths with rapidly changing radius values for two reasons. Firstly, the vesselness filter is noise sensitive, and estimating the radius based solely on the filter output is unreliable. Secondly, vessel radii tend to not change rapidly unless branching or abnormalities such as aneurysms occur. By incorporating this cost $C_R(q,p) = |r - r'|/(r_{\max} - r_{\min})$ to the incremental cost function, the vessel width in the segmentation result is rendered smoother with gradually changing radius values (Figure 23b). Here, r_{max} and r_{min} are the maximum and minimum values r can take. In the case where aneurysms or other causes of rapid vessel radius change, an extra seedpoint can be placed at that location to force the optimal medial's radius to correctly adapt.

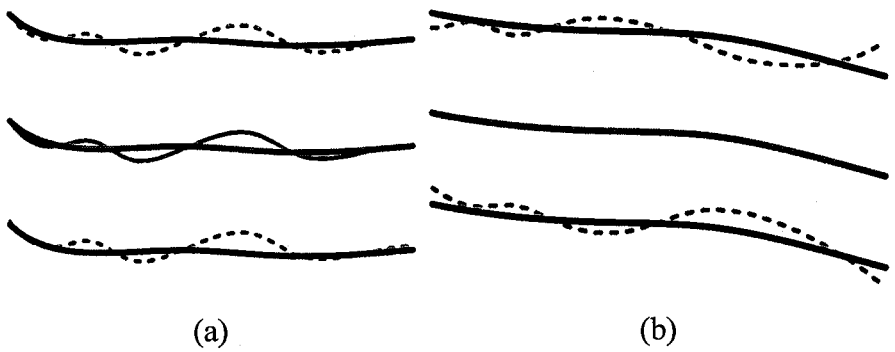


Figure 23. Difference in effects of radial and spatial smoothing. Unsmoothed elements are in gray (solid line = medial, dotted line = boundaries), new medials are in red, and new boundaries are in blue. (a) While vessel width was constant, favouring shorter medial axes results in a smoother medial and vessel boundaries. (b) While the medial was already smooth, minimizing change in radius results in smoother boundary contours.

3.4 Results and Discussions

In this section, we validate the performance of Live-Vessel on synthetic data as well as real medical retinal image data from the publicly available Digital Retinal Images for Vessel Extraction (DRIVE) database [102]. We also provide comparisons to other

state-of-the-art vessel segmentation techniques, as well as demonstrate Live-Vessel's robustness to parameter sensitivity and increasing amounts of noise.

3.4.1 Performance Criteria

We benchmarked the performance of Live-Vessel using the three main criteria typically used to evaluate segmentation results; accuracy, reproducibility, and efficiency [9]. Reproducibility was measured by performing several segmentation trials of the same task, which differed in seedpoint selection, since, realistically, different seedpoints would be chosen by different users or the same user during different trials. Efficiency was calculated as the fraction of the time a user needed to complete the segmentation task using Live-Vessel versus manual tracing. To obtain manual tracing task times, we segmented each image using a simple paint tool, and the task times were similar to those reported in [102].

Accuracy and reproducibility were reported using the Dice similarity metric (which captures the agreement in pixel labels) between the segmentation mask and the golden truth or between resulting segmentation masks in different trials. In retinal vessel segmentation, blood vessel boundaries are oftentimes unclear and thin vessels are delineated differently, if at all, by different observers. Therefore, for evaluating the segmentation of real retinal images, we calculated its similarity to each manual tracing and compared that to how similar the manual tracings are to each other. Dice similarity incorporates both true and false positives (*TP* and *TN*), and is defined as $C_{Dice} = 2area_{sim}/(area_A + area_B)$. C_{Dice} is the Dice similarity coefficient and $area_{sim}$ is

the intersection between segmentation areas of trials A and B ($area_A$ and $area_B$ respectively). Dice similarity coefficients were averaged over multiple trials.

To compare with other current state-of-the-art vessel segmentation techniques, the widely used accuracy measure $Acc = (TP + TN)/(P + N)$, averaged over tested images, was used. TP and TN here are the sums of true positive and negative pixels in the images' circular field of view ($P + N$ total pixels). While we felt reporting this measure is necessary for comparing Live-Vessel to other reported quantitative results (using the same reporting metric), we believe this does not provide a true measure of accuracy, especially for thin vessels. Since background pixels typically greatly outnumber vessel pixels in an image (by approximately 10 to 1) in the field of view, this accuracy measure is inherently inflated and, we believe, is not the best measure of accuracy in such applications of segmentation (for example, Acc give 90% accuracy even if not a single pixel is labeled as vessel, in a typical image where only 10% of the pixels are vessel pixels).

We note that since 2D medical images suffer from partial area effects, our proposed method's segmentation result uses a fuzzy membership for each pixel, evaluating the fraction of the pixel inside/outside the vessel. Note that the boundary of the vessel obtained via Live-Vessel is calculated with sub-pixel accuracy, being a distant r away from and normal to the vessel medial (Section 3.2). If crisp binary segmentation is desired, a pixel is assigned to the object if this ratio is ≥ 0.5.

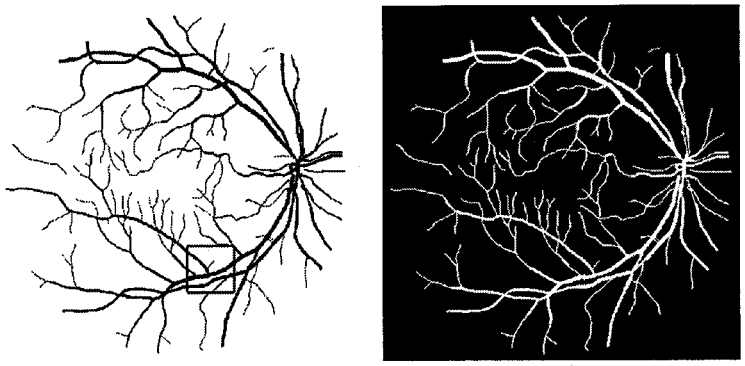
3.4.2 Synthetic Data Segmentation

The manually-traced segmentation masks provided by the DRIVE database were used as realistic 'noise-free' synthetic data. Varying levels of noise were subsequently added. Three randomly selected manual tracings of observer 2 (images 4, 9, 10) from the DRIVE database were each segmented twice, as illustrated in Figure 24. This synthetic data does not suffer from partial area effects, but we found that binary vessel masks are inherently jagged and the masks themselves possessed numerous flaws. One of these is the presence of slight 'indentations' in the vessels that do not appear in the original data. While these flaws were unintentional, our proposed method resolves these issues by encouraging spatial and radial smoothness (Section 3.3.4). We found that vessels with 1-pixel wide widths were segmented correctly (see also Section 3.4.3).

Table 4 summarizes our method's reproducibility and accuracy rates on synthetic data, averaged over 2 trials per image. We found that the reproducibility rate was consistently high and consistent with traditional Livewire [26], reflecting that the locations of the sparse seedpoints only marginally impact the optimal contour in between. We found the accuracy results were also high and consistent (Table 4).

Table 4. Summary of Live-Vessel's performance on 3 random manual tracings in the DRIVE database. Results are shown as averages and standard deviation over 2 trials.

	Dice similarity	**Reproducibility**
Image 04	0.875 ± 0.006	0.983
Image 09	0.849 ± 0.010	0.972
Image 10	0.890 ± 0.008	0.966
Average	**0.871 ± 0.008**	**0.974**

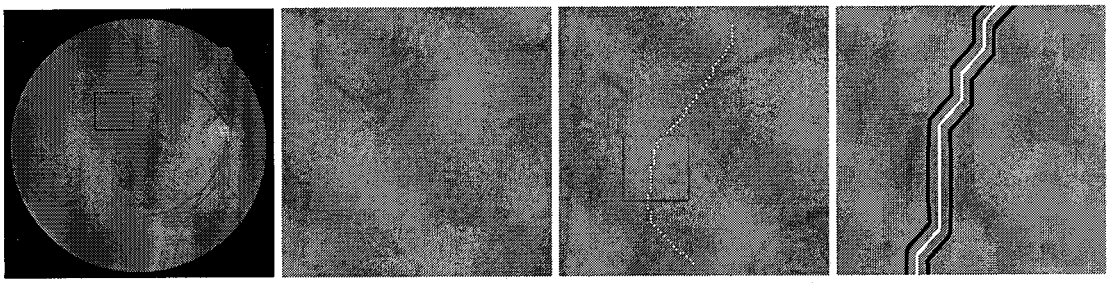


(a) (b)

Figure 24. Synthetic data segmentation. (a) Synthetic Image of retina. (b) Segmentation result using Live-Vessel.

3.4.3 Retinography Data Segmentation

Ten randomly selected retinal images from the DRIVE database were segmented using Live-Vessel to demonstrate our method's performance on real medical vascular images. Test images 1, 3, 4, 5, 6, 7, 9, 10, 19, 20 were ultimately chosen. Example qualitative results are shown in Figure 26. Figure 25 illustrates our proposed method's ability to segment 1-pixel wide vessels with sub-pixel boundary accuracy on real vascular image data.



(a) (b) (c) (d)

Figure 25. Segmentation of 1-pixel wide vessels using Live-Vessel. (a) Original image. (b) Zoomed (red square in (a)) area containing 1-pixel wide vessels. (c) Computed optimal medial path for one of the vessels in (b). (d) Further zoomed view of vessel (green square in (c)). White line denotes optimal medial axis, black lines indicate the optimal vessel boundaries.

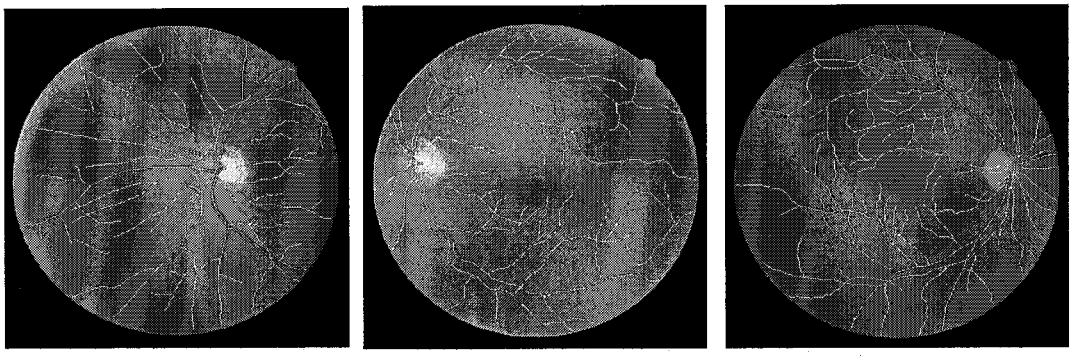
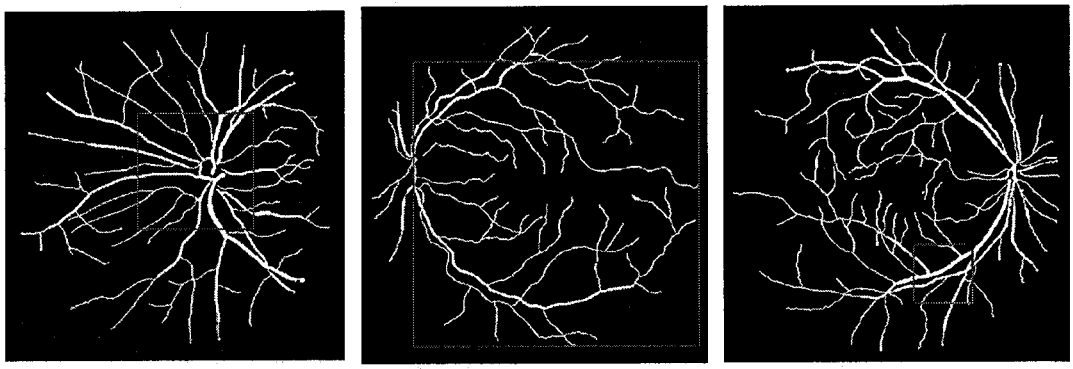
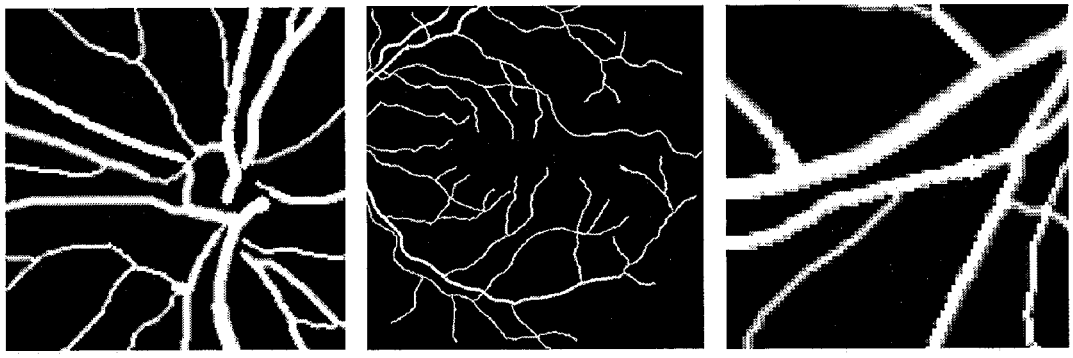
	Figure removed due to copyright permission.	Figure removed due to copyright permission.	Figure removed due to copyright permission.
	Retinal image #4 from	Retinal image #9 from	Retinal image #10 from
(a)	M. Niemeijer, J.J. Staal, B. van Ginneken, M. Loog, and M.D. Abramoff, Comparative Study of Retinal Vessel Segmentation Methods on a new Publicly Available Database, SPIE Medical Imaging (2004) 648-656.	M. Niemeijer, J.J. Staal, B. van Ginneken, M. Loog, and M.D. Abramoff, Comparative Study of Retinal Vessel Segmentation Methods on a new Publicly Available Database, SPIE Medical Imaging (2004) 648-656.	M. Niemeijer, J.J. Staal, B. van Ginneken, M. Loog, and M.D. Abramoff, Comparative Study of Retinal Vessel Segmentation Methods on a new Publicly Available Database, SPIE Medical Imaging (2004) 648-656.
(b)			
(c)			
(d)			

Figure 26. Sample results using our proposed method. For each column: (a) Original retinal image. (b) Optimal medial axis computed by Live-Vessel (white curve) overlaid on original image. (c) Live-Vessel segmentation mask. (d) Close-up of segmentation mask.

Table 5. Summary of the average accuracy and reproducibility results of Live-Vessel (LV) compared to manual traces (MT1 and MT2), reporting averages and standard deviation over two trials. Dice similarity is the reported measure. Image size is 565 × 584 pixels throughout.

	Segmentation Accuracy			LV Segmentation Reproducibility
	MT1 vs. MT2	**LV vs. MT1**	**LV vs. MT2**	
Image 01	0.804	0.778 ± 0.023	0.810 ± 0.021	0.969
Image 03	0.785	0.784 ± 0.011	0.760 ± 0.008	0.972
Image 04	0.802	0.780 ± 0.021	0.794 ± 0.038	0.984
Image 05	0.790	0.744 ± 0.003	0.762 ± 0.008	0.976
Image 06	0.770	0.789 ± 0.006	0.815 ± 0.012	0.954
Image 07	0.768	0.729 ± 0.004	0.764 ± 0.025	0.966
Image 09	0.800	0.792 ± 0.032	0.786 ± 0.017	0.959
Image 10	0.787	0.779 ± 0.004	0.799 ± 0.024	0.981
Image 19	0.825	0.765 ± 0.005	0.811 ± 0.015	0.967
Image 20	0.770	0.728 ± 0.009	0.821 ± 0.006	0.956
Average	**0.790**	**0.767 ± 0.012**	**0.792 ± 0.017 → higher than MT1 vs. MT2**	**0.968**

Quantitative results indicated high reproducibility rates, similar to our results on synthetic data. Accuracy was found to be similar to both manual segmentations, and this similarity was comparable to that of manual tracings to each other. In some cases, Live-Vessel had a higher pixel agreement with either or both manual tracings than between the two manual tracings themselves. We also found that Live-Vessel had higher similarity results for the second manual tracing (Table 5) than the first tracing in 8 out of 10 images. We believe that in addition to user fatigue, there may be perceptive differences between DRIVE's different users, perhaps due to computer hardware such as display quality. Figure 27 illustrates the differences between these users, computed using the 'exclusive or' operation. Variability is evident for both thick and thin vessels. Table 6 summarizes

our findings on Live-Vessel's efficiency tests. By only requiring the user to guide the medial path of vessels, we found that segmentation using Live-Vessel on average required less than 25% of the time it takes to segment using manual tracing (75% reduction). Images with more vessels or with vessels that are not as clear expectedly required more time to segment.

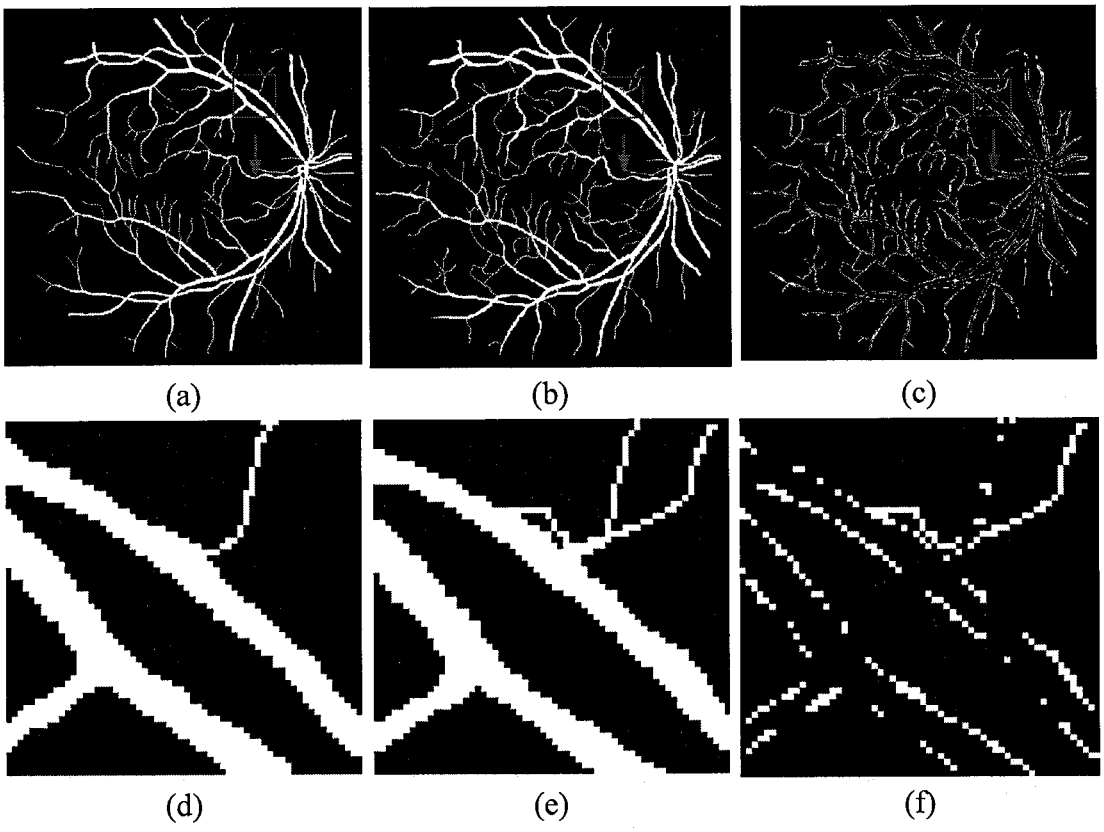


Figure 27. Manual segmentation differences between experts. (a) Observer 1. (b) Observer 2. (c) Difference between (a) and (b). (d-e) Close-up of (a) and (b) respectively. (f) Difference between (d) and (e).

Table 6. Efficiency results of our proposed Live-Vessel (LV) compared to a manual trace (MT), reporting average and standard deviation over two trials. Efficiency was measured as the fraction of the manual task time needed to generate the mask. Results shown are averages of two trials, and are reported in seconds. The images size is 565 × 584 pixels. Average reduction in time is 75.3%.

Segmentation Efficiency			
	MT	**LV**	**LV/MT (%)**
Image 01	6154	1260 ± 48	20.5 ± 0.8
Image 03	6689	1683 ± 98	25.2 ± 1.5

Image 04	6839	1632 ± 8	23.9 ± 0.1
Image 05	7613	2052 ± 132	26.9 ± 1.7
Image 06	5741	1437 ± 33	25.0 ± 0.6
Image 07	6488	2042 ± 67	31.5 ± 1.0
Image 09	5546	1382 ± 52	24.9 ± 0.9
Image 10	5394	1111 ± 52	20.6 ± 1.0
Image 19	4877	1316 ± 57	27.0 ± 1.2
Image 20	6807	1490 ± 87	21.9 ± 1.3
Average	**6215**	**1541 ± 63**	**24.7 ± 1.0 → 75.3% reduction**

Finally, we also quantitatively compared Live-Vessel's performance to the latest reported techniques in 2D retinography segmentation. In terms of measured accuracy, our proposed technique is quite similar (Table 7), although as described previously, we believe this accuracy measure is biased in all cases (as background pixels significantly outnumber vessel pixels in all images). It is important to note that none of the other reported techniques ensures topological integrity and thus may result in many broken vessels and spurious (non-vessel like) detected structures (e.g. Figure 11a in [70], Figure 6 in [72], Figure 6 in [71], and Figure 15 in [74]). Thin vessels (1-2 pixel wide), which can constitute a large portion of a vessel tree's total length, but may not contribute much to the accuracy calculation, are especially problematic in reported methods; however, they are handled well by Live-Vessel. Additionally, Live-Vessel computes the optimal medial axes with radius values along the vessel (in addition to the boundaries and segmentation masks), which simplifies subsequent vessel tree analysis (e.g. by avoiding vessel skeletonization).

Table 7. Accuracy comparison of state-of-the-art methods and the proposed method Live-Vessel, using the widely used accuracy assessment method.

Technique	Accuracy
Jiang *et al.*	0.9337
Ricci and Perfetti	0.9595
Lam and Yan	0.9474
Soares *et al.*	0.9466
Staal *et al.*	0.9441
Live-Vessel	**0.9460**

3.4.4 Live-Vessel Robustness and Parameter Selection

We performed detailed sensitivity analysis in order to determine the extent of change in accuracy (Dice similarity) when varying the weighting parameters using in our optimization function the incremental cost function. Tests were done on observer 2's manual tracing for image 10 in the DRIVE database. Our implementation uses default equal weighting ($w_{1..5}=1$) for each cost term weight in the incremental cost function. Results of changing the weight of each term are shown in Table 8. It can be seen that even with ±50% variability in each parameter, Live-Vessel's accuracy was stable, with an average magnitude in percentage change of 2.44%, with a maximum of 7.781%, and a minimum of 0%. This confirms d Live-Vessel's insensitivity to significant parameter variations.

Table 8. Effect of parameter selection on Dice similarity (accuracy) between Live-Vessel and each manual tracing (MT1, MT2). Percentage of accuracy change provided for each experiment. Weighting for vesselness (V), vessel direction consistency (Ev), image evidence (Ie), radius smoothness (R), and spatial smoothness (XY) were raised or lowered (by ±50%).

Changing weights in the incremental cost function	Accuracy and % change			
	vs. MT1	vs. MT2	vs. MT1	vs. MT2
	-50% $w_i=0.5$	-50% $w_i=0.5$	+50% $w_i=1.5$	+50% $w_i=1.5$
No change	0.784	0.760	0.784	0.760

($w_{1..5} = 1$)				
V	0.723 (-7.781%)	0.738 (-2.895%)	0.779 (-0.638%)	0.792 (4.211 %)
Ev	0.764 (-2.551%)	0.755 (-0.658%)	0.795 (1.403%)	0.762 (0.263 %)
Ie	0.751 (-4.209%)	0.748 (-1.579%)	0.814 (3.827 %)	0.759 (-0.132%)
R	0.735 (-6.250%)	0.729 (-4.079%)	0.791 (0.893%)	0.760 (0.000 %)
XY	0.775 (-1.148%)	0.774 (1.842 %)	0.762 (-2.806%)	0.773 (1.711%)

To test Live-Vessel's robustness to noise, observer 2's manual tracing for image 10 in the DRIVE database was subjected to incremental levels of AWGN and then segmented using consistent seedpoints in Live-Vessel, but only on vessels still visible to the observer (given the progressing noise levels). Accuracy levels were then plotted against the peak signal to noise ratio (PSNR) of the volume, defined as $PSNR = 20\log_{10}(\max(ObjectIntensity))/\sigma_{noise}$. The segmentation results are illustrated in Figure 28. As expected, we found that as noise increased, segmentation speed was naturally reduced because denser sets of seedpoints were needed to maintain accuracy. Under high noise, the eigenvector consistency, radial smoothness, and spatial smoothness costs were especially important, as vesselness response and image evidence measures were adversely affected. Our tests show that even under high noise (PSNR = -8.11 dB in Figure 28d), our proposed method was still successfully tracking the medial axes of reasonably large vessels.

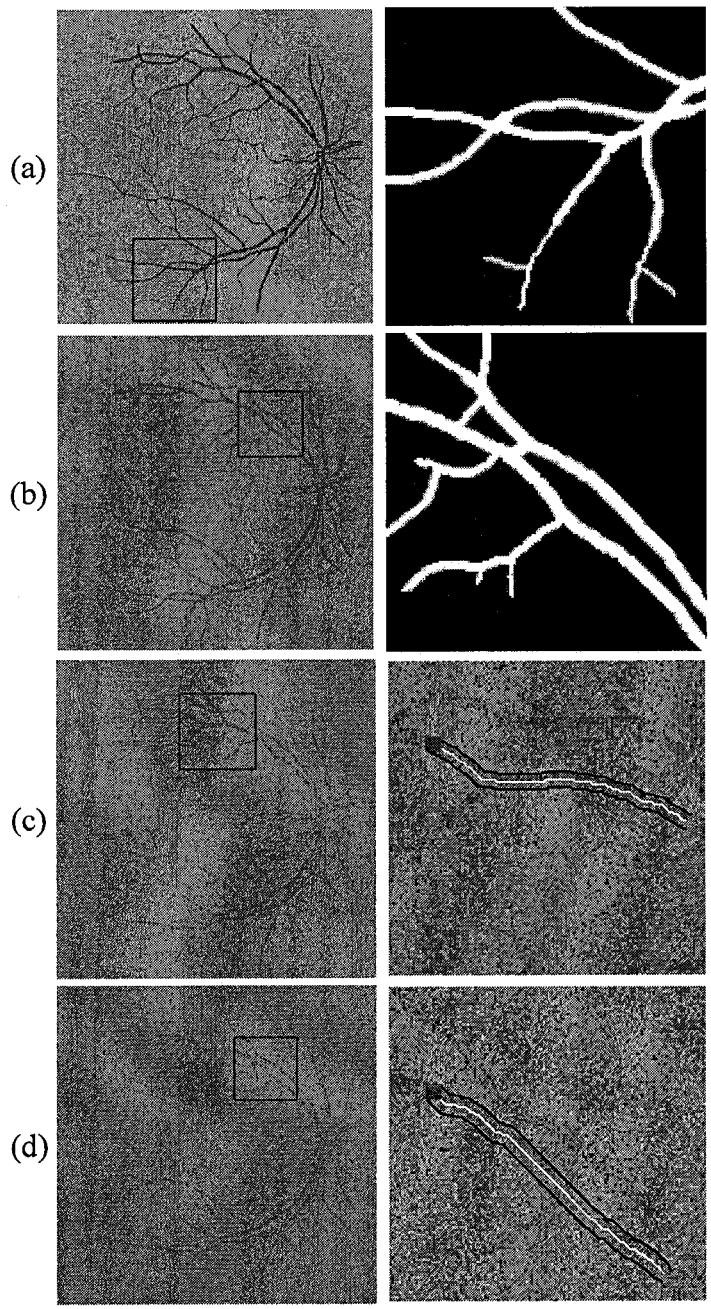


Figure 28. Segmentation of a synthetic image under increasing AWGN noise. Red squares in the first column denote the field of view zoomed in the second column. (a) PSNR = 27.73 dB. (b) PSNR = 10.22 dB. (c) PSNR = 0 dB. (d) PSNR = -8.11 dB. In the right column of (a) and (b) segmentation masks are shown. In the right column of (c) and (d), black contours denote detected boundary points. White contours represent optimal medial axes from a seedpoint (green cross) to the next point (solid red square).

Chapter 4

Conclusions

Medical imaging poses many problems to structural segmentation. Although much research is put into this field, it is still difficult to use computation alone to solve these issues without incorporation of human insight to recognize and locate the object of interest and to differentiate between a good and bad segmentation result.

4.1 Contributions

The two highly-automated, interactive frameworks that we presented here are efficient in melding the advantages of both extremes of the fully-automated to manual segmentation spectrum. Our contributions here are two interactive, globally optimal techniques [103][104]. Both our 3D Livewire method and Live-Vessel have been submitted as journal submissions after further refinement and testing.

1. 2D to 3D Livewire extension for 3D objects:

   M. Poon, G. Hamarneh, and R. Abugharbieh, "Segmentation of Complex Objects with Non-spherical Topologies from Volumetric Medical Images using 3D Livewire," SPIE Medical Imaging 6512-31, pp. 1–10, 2007.

   - Handles arbitrary topologies, protrusions, concavities, and branching
   - Integrated into a user-friendly segmentation tool
   - Accurate, highly reproducible, and efficient compared to traditional Livewire
   - Robust to high levels of noise and insensitive to parameter variability

2. Live-Vessel for segmenting 2D retinography images:

   M. Poon, G. Hamarneh, R. Abugharbieh, Live-Vessel: Extending Livewire for Simultaneous Extraction of Optimal Medial and Boundary Paths in Vascular Images, MICCAI (2007) 444-451.

   - Simultaneously extracts vascular medial paths and boundaries
   - Optimizes over spatial (x,y) and non-spatial (*radius*) variables
   - Optimizes for vesselness, vessel direction consistency, edge information, and radial and spatial smoothness
   - Comparable accuracy to manual segmentation, reproducible, highly efficient compared to manual segmentation
   - Robust to high levels of noise and insensitive to parameter variability

4.2 Limitations and Future Directions

Despite the proposed methods' benefits, there are several limitations and areas for improvement.

Extensibility to Other Modalities

For instance, their extensibility to other data types is yet to be proven. Other modalities such as diffusion tensor imaging, ultrasound, or magnetic resonance angiography pose different problems than MRI or CT, which require implementation specifics that cater to these issues. Also, while 2D and 3D image modalities are common, medical images can have higher dimensionalities, for example, the incorporation of the time dimension to capture structural deformities or brain function changes.

Sub-pixel Accuracy

Another limitation of our approaches is that they do not offer sub-pixel accuracy, as the contours in our 3D Livewire approach and the medial paths in Live-Vessel lies on pixel centres. While this is less of an issue in high resolution images, medical images oftentimes have low resolution in one or more dimensions. One possible way to accomplish this is to use refinement subdivision procedures such as [105] or up-sample the image prior to segmentation. Also, as an option, deformable models [12] may be used to refine the segmentation at the expense of user-control and predictability.

3D Livewire - Algorithm Robustness to User Errors

With regard to the 3D Livewire method, the more immediate goals include enhancing algorithm robustness to user-mistakes. While Livewire contours is adept in snapping to object edges, incorrect input, even if it is off by one or two pixels, creates small protrusions or indentations which can affect the quality of the final result. A possible solution to this may be an option to allow our segmentation tool to search within its neighbouring pixels to find a more suitable seedpoint. We believe this should remain as a user option because we feel the user should be able to assert full control of seedpoint locations if necessary.

3D Livewire - Parallelization

Another potential improvement is that although graph search speed is not an issue with traditional 2D Livewire, our method's need to perform a large number of automated Livewire operations on unseen slices can benefit from speed-up algorithms. One possible method is to parallelize Dijkstra's algorithm on multiple processors [108]. Also, since the segmentation of an unseen slice is not dependent on adjacent slices, multiple processors can linearly decrease the automated phase of our proposed method.

3D Livewire - Relationship Between Topology and Required User-guided Contours

Another limitation is that users currently require initial practice and understanding of the properties of the structure they wish to segment, and place user-guided contours on select slices. This is a concern because biological structures vary widely in size, shape, and topology, and we wish to refine this method such that non-experts with minimal

training can perform segmentation with reasonable results. Therefore, we believe that this project's future direction should include the understanding of the relationship between the amount and orientation of user-guided contours necessary for a correct segmentation and the shape, size, and topology of the object. A possible approach to solve this problem is to apply Art Gallery Theorems [106], which applies for complex topology and 3D extensions. Possible benefits include being able to automatically determine and suggest to the user the locations of slices to place manually-guided Livewire contours, cutting down on segmentation task time and improving reproducibility.

3D Livewire - Oblique Slices

In addition, we would like to look into how oblique slices, rather than just orthogonal slices, can be used to initiate this 3D Livewire algorithm, because oblique slices may characterize an object more accurately. The main challenge will be to reformulate the turtle algorithm such that non-orthogonal 'tracks' can be traversed.

3D Livewire - Training Parameters

While we would like to maintain our algorithm's flexibility over as many modalities of medical images as possible, we believe that training the weights of the optimization terms to suit a particular modality would benefit the accuracy and usability of the segmentation. This may be obtained be collecting a set of training image-segmentation pairs, and optimizing for the weights that give delineations as close as possible to the ground-truth training segmentation.

Livewire for Globally Optimal Surfaces

Lastly, another future direction may be to extend Livewire from finding an optimal 1D contour in 2D (or even 3D space) to finding globally optimal 2D surfaces such as those found in [86]. An idea is to use 1D Livewire contours along the surface of an object in 3D space and iteratively partition the object surface for polygon generation. Another potential method is to define surface 'patches' along an object and using Livewire to determine an optimal path along these 'surface nodes', thus performing a 3D segmentation problem using a 2D graph search. This approach would be similar to [107], but without the need for pre-segmentation.

Live-Vessel Efficiency

For Live-Vessel, the immediate improvements we will pursue include the reduction of the computation time required for the graph search. The reason is that our 2D+R graph search algorithm is exponentially more computational than a 2D graph search, with each node currently having 32 neighbors compared to 8 for the 2D case. We are again looking into parallelization [108], as well as limiting the graph search operation similarly to [34][35].

Live-Vessel for 3D Vessel Segmentation

We believe Live-Vessel is well-suited for 2D retinography images, but the next natural step would be to extend this framework even further to (x,y,z,r) for segmentation of 3D vessels such as ones found using magnetic resonance. One of the design challenges here will be to reduce the graph search as mentioned above, as Dijkstra's algorithm in 4D

would be even more computationally demanding. Also, in any interactive 3D segmentation schemes, human computer interaction such as visualization and input method becomes more complex. The goal here will be to develop an interaction scheme that allows users to see the proposed vessel before they validate it. While rendering 3D vessels is not difficult, users oftentimes judge the quality of a segmentation contour based on intensity information from its surrounding pixels/voxels; thus this information will have to be incorporated into the 3D vessel visualization as well.

Further Live-Vessel Validation

Also, in this thesis, we benchmarked our method's accuracy on two different measures; however, adopting a different validation method such as [109] that evaluates the medial itself would benefit the use of Live-Vessel in vessel tree analyses and vessel tracking. Similar to this idea, while the DRIVE database only provides two sets of expert segmentations, it would be beneficial to investigate exactly why these differ and to derive a ground truth from these observers or by using additional experts.

Bibliography

[1] A. Uthama, R. Abugharbieh, A. Travoulsee, and M. McKeown, Invariant SPHARM Shape Descriptors for Complex Geometry in MR Region of Interest Analysis, IEEE EMBS (2007) 1322-1325.

[2] A. Convit, M. Leon, C. Tarshish, S. Santi, W. Tsui, H. Rusinek, and A. George, Specific Hippocampal Volume Reductions in Individuals at Risk for Alzheimer's Disease, Neurobiology of Aging (1997) 131-138.

[3] E. Bullitt, G. Gerig, S. Aylward, S. Joshi, K. Smith, M. Ewend, and W. Lin, Vascular Attributes and Malignant Brain Tumors, MICCAI (2003) 671-679.

[4] J. Ross, T. Masaryk, M. Modic, P. Ruggieri, E. Haacke, W. Selman, Intracranial Aneurysms: Evaluation by MR Angiography, American Journal of Neuroradiology 11(3) (1990) 449-455.

[5] F. Skovborg, A. Nielsen, E. Lauritzen, and O. Hartkopp, Diameters of the Retinal Vessels in Diabetic and Normal Subjects, Diabetes (1969) 292-298.

[6] T. Wong and R. McIntosh, Hypertensive Retinopathy Signs as Risk Indicators of Cardiovascular Morbidity and Mortality, British Medical Bulletin (2005) 57-70.

[7] S. Aylward, J. Jomier, S. Weeks, and E. Bullitt, Registration and Analysis of Vascular Images, International Journal of Computer Vision (2002) 123-138.

[8] J. Rajapakse and F. Kruggel, "Segmentation of MR images with intensity inhomogeneities," Image and Vision Computing 16, pp. 165–180, 1998.

[9] S. Olabarriaga and A. Smeulders, "Interaction in the segmentation of medical images: A survey," Medical Image Analysis 5, pp. 127–142, 2001.

[10] R. Gonzalez and R. Woods, Digital Image Processing *2nd Ed.*, Prentice-Hall Inc., Upper Saddle River, New Jersey, USA, 2002.

[11] T. Yoo (Editor), Insight into Images, A K Peters Ltd., Wellesey, MA, USA, 2004.

[12] T. McInerney and D. Terzopoulos, "Deformable models in medical image analysis: A survey," Medical Image Analysis 1, pp. 91–108, 1996.

[13] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active contour models," International Journal of Computer Vision 1, pp. 321–331, 1988.

[14] J. Liang, T. McInerney, and D. Terzopoulos, "United snakes," Medical Image Analysis 10, pp. 215–233, 2006.

[15] S. Osher and N. Paragios, Geometric Level Set Methods in Imaging Vision and Graphics, Springer Verlag, 2003.

[16] V. Caselles, R. Kimmel, and G. Sapiro, "Geodesic active contours," IJCV 22(1), pp. 61–79, 1997.

[17] J. A. Sethian, Level Set Methods; Evolving Interfaces in Geometry, Fluid Mechanics, Cambridge University Press, 1996.

[18] Y. Kang, K. Engelke, and W. Kalender, "Interactive 3D editing tools for image segmentation," Medical Image Analysis 8, pp. 35–46, 2004.

[19] T. Buck, H. Ehricke, W. Strasser, and L. Thurfjel, 3D Segmentation of Medical Structures by Integration of Ray-casting with Anatomic Knowledge, Computers and Graphics (1995) 441-449.

[20] J. Cabral, K. White, Y. Kim, and E. Effmann, Interactive Segmentation of Brain Tumors in MR Images using 3D Region-Growing, SPIE Medical Imaging (1993) 171-181.

[21] J. Udupa, L. Wei, S. Samarasekera, M. Van Buchem, and R. Grossman, Multiple Sclerosis Lesion Quantification using Fuzzy-Connectedness Principles, IEEE TMI (1997) 598-609.

[22] Y. Boykov and G. Funka-Lea, "Graph cuts and efficient N-D image segmentation," International Journal of Computer Vision 70, pp. 109–131, 2006.

[23] W. Higgins, J. Reinhardt, and W. Sharp, Semi-automatic Construction of 3D Medical Image-Segmentation Processes, SPIE Medical Imaging (1994) 59-71.

[24] K. Hinshaw, R. Altman, and J. Brinkley, Shape-based Models for Interactive Segmentation of Medical Images, SPIE Medical Imaging (1995) 771-780.

[25] T. McInerney and S. Akhavan, "Sketch initialized snakes for rapid, accurate and repeatable interactive medical image segmentation," ISBI , pp. 398–401, 2006.

[26] W. Barrett and E. Mortensen, "Interactive live-wire boundary extraction," Medical Image Analysis 1, pp. 331–341, 1997.

[27] O. Wink, W. Niessen, and M. Viergever, Multiscale Vessel Tracking, IEEE TMI (2004) 130-133.

[28] C. Rother, V. Kolmogorov, and A. Blake, "GrabCut: interactive foreground extraction using iterated graph cuts," ACM Transactions on Graphics , pp. 309–314, 2004.

[29] L. Grady, "Random walks for image segmentation," IEEE Transactions Pattern Analysis and Machine Intelligence 28, pp. 1768–1783, November 2006.

[30] R. Bellman, Dynamic Programming, Princeton University Press, 1957.

[31] E. Dijkstra, "A note on two problems in connexion with graphs," In Numerical Mathematik 1, pp. 269–270, December 1959.

[32] J. Sethian, Level Set Methods and Fast Marching Method: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science, Cambridge University Press, 1999.

[33] J. Canny, "A computational approach to edge detection," IEEE Transactions Pattern Analysis and Machine Intelligence 8(6), pp. 679–698, 1986.

[34] A. Falcao, J. Udupa, S. Samarasekera, S. Sharma, B. Hirsch, and R. Lotufo, "User-steered image segmentation paradigms: Live wire and live lane," Graphical Models and Image Processing 60, pp. 233–260, July 1998.

[35] A. Falcao, J. Udupa, and F. Miyazawa, "An ultra-fast user-steered image segmentation paradigm: live wire on the fly," IEEE TMI 19, pp. 55–62, Jan 2000.

[36] H. W. Kang and S. Y. Shin, "Enhanced lane: interactive image segmentation by incremental path map construction," Graphical Models 64(5), pp. 282–303, 2002.

[37] A. Schenk, G. Prause, and H. Peitgen, "Local cost computation for efficient segmentation of 3D objects with live wire," SPIE Medical Imaging , pp. 1357–1364, 2001.

[38] T. McInerney and D. Terzopoulos, "T-snakes: Topology adaptive snakes," Medical Image Analysis 4, pp. 73–91, 2000.

[39] H. Delingette, "General object reconstruction based on simplex meshes," IJCV 32, pp. 111–146, 1999.

[40] J.-O. Lachaud and A. Montanvert, "Deformable meshes with automated topology changes for coarse-to-fine 3D surface extraction," Medical Image Analysis 3(2), pp. 187–207, 1998.

[41] J. Montagnat, H. Delingette, and N. Ayache, "A review of deformable surfaces: Topology, geometry and deformation," 19(14), pp. 1023–1040, 2001.

[42] X. Bresson, S. Esedoglu, P. Vandergheynst, J.-P. Thiran, and S. Osher, "Fast global minimization of the active contour/snake model," Journal of Mathematical Imaging and Vision 28(2), (2007) 151-167.

[43] L. Cohen and R. Kimmel, "Global minimum for active contour models: A minimal path approach," IJCV 24, pp. 57–78, 1997.

[44] N. Paragios, "User-aided boundary delineation through the propagation of implicit representations," MICCAI , pp. 678–686, 2003.

[45] M. Rumpf and R. Strzodka, "Level set segmentation in graphics hardware," IEEE international Conference on Image Processing , pp. 1103–1106, 2001.

[46] A. Lefohn, J. Cates, and R. Whitaker, "Interactive, GPU-based level sets for 3D segmentation," MICCAI , pp. 564–572, 2003.

[47] P. Yushkevich, J. Piven, H. Hazlett, R. Smith, S. Ho, J. Gee, and G. Gerig, "User-guided 3D active contour segmentation of anatomical structures: Significantly improved efficiency and reliability," NeuroImage 31, pp. 1116–1128, 2006.

[48] M. Urschler, H. Mayer, R. Bolter, and F. Leberl, "The livewire approach for the segmentation of left ventricle electron-beam CT images," In 26th Workshop of the Austrian Association for Pattern Recognition (AGM/AAPR) , 2002.

[49] A. Gougoutas, A. Wheaton, A. Borthakur, E. Shapiro, J. Kneeland, J. Udupa, and R. Ravinder, "Cartilage volume quantification via live wire segmentation," Academic Radiology 11, pp. 1389–1395, Jan 2004.

[50] H. Zhang, J. Nosher, and P. Yim, "Surface reconstruction from orthogonal contours," SPIE Medical Imaging , pp. 98–104, 2006.

[51] M. Jackowski, M. Satter, and A. Goshtasby, "Approximating digital 3D shapes by rational gaussian surfaces," IEEE Transactions on Visualization and Computer Graphics 9(1), pp. 56–69, 2003.

[52] H. Zhao, S. Osher, B. Merriman, and M. Kang, "Implicit, nonparametric shape reconstruction from unorganized points using a variational level set method," Computer Vision and Image Understanding 80, pp. 295–319, 2000.

[53] G. Turk and J. O'Brien, "Shape transformation using variational implicit functions," Proceedings of SIGGRAPH 99: Computer Graphics , pp. 335–342, July 1999.

[54] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," in SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques, pp. 71–78, ACM Press, (New York, NY, USA), 1992.

[55] R. Saboo, J. Rosenman, and S. Pizer, "Geointerp: Contour interpolation with geodesic snakes," The Insight Jounal , July 2006.

[56] Y. Boykov and M. Jolly, "Interactive organ segmentation using graph cuts," MICCAI , pp. 276–286, 2000.

[57] A. Souza, J. Udupa, G. Grevera, D. O. Y. Sun, N. Suri, and S. M, "Iterative live wire and live snake: New user-steered 3D image segmentation paradigms," SPIE Medical Imaging , pp. 1159–1165, March 2006.

[58] A. Schenk, G. Prause, and H. Peitgen, "Efficient semiautomatic segmentation of 3D objects in medical images," MICCAI , pp. 186–195, 2000.

[59] F. Malmberg, E. Vidholm, and I. Nystrom, A 3D Live-Wire Segmentation Method for Volume Images Using Haptic Interaction.

[60] A. Falcao and F. Bergo, "Interactive volume segmentation with differential image foresting transforms," IEEE TMI 23, pp. 1100–1108, September 2004.

[61] A. Falcao and J. Udupa, "A 3D generalization of user-steered live-wire segmentation," Medical Image Analysis 4, pp. 389–402, 2000.

[62] K. Lu and W. Higgins, "Improved 3D live-wire method with application to 3D ct chest image analysis," SPIE Medical Imaging , pp. 189–203, 2006.

[63] G. Hamarneh, J. Yang, C. McIntosh, and M. Langille, "3D live-wire-based semiautomatic segmentation of medical images," SPIE Medical Imaging , pp. 1597-1603, 2005.

[64] S. Papert, Mindstorms: Children, Computers, and Powerful Ideas, Basic Books, Inc., New York, NY, USA, 1980.

[65] M. Ackerman, "The visible human project," Proceedings of the IEEE 86, pp. 504–511, March 1998.

[66] A. Szymczak, A. Stillman, A. Tannenbaum, and K. Mischaikow, Coronary Vessel Trees from 3D Imagery: A Topological Approach, Medical Image Analysis 10(4) (2004) 548-559.

[67] A. Can, H. Shen, J. N. Turner, H. L. Tanenbaum, and B. Roysam, Rapid Automated Tracing and Feature Extraction from Live High-resolution Retinal Fundus Images using Direct Exploratory Algorithms, IEEE Transactions on Information Technology in Biomedicine (1999) 125-138.

[68] J. Lowell, A. Hunter, D. Steel, A. Basu, R. Ryder, and R.L. Kennedy, Measurement of Retinal Vessel Widths From Fundus Images Based on 2-D Modeling, IEEE TMI (2004) 1196-1204.

[69] C. Kirbas and F. Quek, A Review of Vessel Extraction Techniques and Algorithms, ACM Computing Surveys (2004) 81-121.

[70] E. Ricci and R. Perfetti, Retinal Blood Vessel Segmentation Using Line Operators and Support Vector Classification, IEEE TMI (2007) 1357-1365.

[71] J. Soares, J. Leandro, R. Cesar, H. Jelinek, and M. Cree, Retinal Vessel Segmentation Using the 2-D Gabor Wavelet and Supervised Classification, IEEE TMI (2006) 1214-1222.

[72] J. Staal, M. Abramoff, M. Niemeijer, M. Viergever, and B. Ginneken, Ridge-Based Vessel Segmentation in Color Images of the Retina, IEEE TMI (2004) 501-509.

[73] D. Eberley, R. Gardner, B. Morse, S. Pizer, and C. Scharlach, Ridges for Image Analysis, Journal of Mathematical Imaging and Vision (1994) 353-373.

[74] B. Lam and H. Yan, A Novel Vessel Segmentation Algorithm for Pathological Retina Images Based on the Divergence of Vector Fields, IEEE TMI (2008) 237-246.

[75] L. Lorigo, O.D. Faugeras, W. Grimson, R. Keriven, R. Kikinis, A. Nabavi, and C.-F. Westin, CURVES: Curve evolution for vessel segmentation, Medical Image Analysis (2001) 195-206.

[76] R. Manniesing and W. Niessen, Local Speed Functions in Level Set Based Vessel Segmentation, MICCAI (2004) 475-482.

[77] R. Manniesing, M. Viergever, and W. Niessen, Vessel Axis Tracking Using Topology Constrained Surface Evolution, IEEE TMI (2007) 309-316.

[78] D. Nain, A. Yezzi, and G. Turk, Vessel Segmentation Using a Shape Driven Flow, MICCAI (2004) 51-59.

[79] H. Luo, Q. Lu, and R. Acharya, Robust Snake Model, IEEE CVPR (2000) 452-457.

[80] A. Frangi, W. Niessen, R. Hoogeveen, T. van Walsum, and M. Viergever, Quantitation of vessel morphology from 3D MRA, MICCAI (1999) 358-367.

[81] T. McInerney and D. Terzopoulos, Topology Adaptive Deformable Surfaces for Medical Image Volume Segmentation, IEEE TMI 18(10) (1999) 840-850.

[82] A. Vasilevskiy and K. Siddiqi, Flux Maximizing Geometric Flows, IEEE TPAMI (2002) 1565-1578.

[83] P. Yan and A. Kassim, Segmentation of Volumetric MRA Images by using Capillary Active Contour, Medical Image Analysis (2006) 317-329.

[84] T. Chan, S. Esedoglu, and M. Nikoloa, Algorithms for Finding Global Minimizers of Image Segmentation and Denoising Models, SIAM Journal on Applied Mathematics 66(5) (2006) 1632-1648.

[85] N. Hraiech, M. Carroll, M. Rochette, J. Coatrieux, 3D Vascular Shape Segmentation for Fluid-Structure Modeling, SMI (2007) 226-231.

[86] K. Li, X. Wu, D. Chen, and M. Sonka, Optimal Surface Segmentation in Volumetric Images – A Graph-Theoretic Approach, IEEE PAMI 28(1) (2006) 119-134.

[87] T. Koller, G. Gerig, G. Szekely, and D. Dettwiler, Multiscale Detection of Curvilinear Structures in 2-D and 3-D Image Data, ICCV (1995) 864-869.

[88] C. Lorenz, I. Carlsen, T. Buzug, C. Fassnacht, and J. Weese, Multi-scale Line Segmentation With Automatic Estimation of Width, Contrast and Tangential Direction in 2D and 3D Medical Images, CVRMED-MRCAS'97, LNCS (1997) 233-242.

[89] Y. Sato, S. Nakajima, N. Shiraga, H. Atsumi, S. Yoshida, T. Koller, G. Gerig, and R. Kikinis, Three-Dimensional Multi-scale Line Filter for Segmentation and Visualization of Curvilinear Structures in Medical Images, Medical Image Analysis (1998) 143-168.

[90] S. Aylward and E. Bullitt, Initialization, Noise, Singularities and Scale in Height Ridge Traversal for Tubular Object Centerline Extraction, IEEE TMI (2002) 61-75.

[91] A. Frangi, W. Niessen, K. Vincken, and M. Viergever, Multiscale Vessel Enhancement Filtering, MICCAI (1998) 130-137.

[92] C. Canero and P. Radeva, Vesselness Enhancement Diffusion, Pattern Recognition Letters (2003) 3141-3151.

[93] R. Manniesing, M. A. Viergever, and W.J. Niessen, Vessel Enhancing Diffusion: A Scale Space Representation of Vessel Structures, Medical Image Analysis (2006) 815-825.

[94] M. Martinez-Perez, A. Hughes, S. Thom, A. Bharath, and K. Parker, Segmentation of Blood Vessels from Red-free and Fluorescein Retinal Images, Medical Image Analysis (2007) 47-61.

[95] K. Krissian, G. Malandain, N. Ayache, R. Vaillant, and Y. Trousset, Model Based Detection of Tubular Structures in 3D Images, Computer Vision and Image Understanding (2000) 130-171.

[96] C. Wu, G. Agam, and P. Stanchev, A Hybrid Filtering Approach to Retinal Vessel Segmentation, IEEE ISBI (2007) 604-607.

[97] M. Sofka and C. Stewart, Retinal Vessel Centerline Extraction Using Multiscale Matched Filters, Confidence and Edge Measures, IEEE TMI (2006) 1531-1546.

[98] C. McIntosh and G. Hamarneh, Vessel Crawlers: 3D Physically-based Deformable Organisms for Vasculature Segmentation and Analysis, IEEE CVPR (2006) 1084-1091.

[99] T. Deschamps and L. Cohen, Fast extraction of minimal paths in 3D images and applications to virtual endoscopy, Medical Image Analysis (2001) 281-299.

[100] S. Young, B. Movassaghi, J. Weese, and V. Rasche, 3D Vessel Axis Extraction using 2D Calibrated X-ray Projections for Coronary Modeling, SPIE Medical Imaging (2003) 1491-1498.

[101] H. Li and A. Yezzi, Vessels as 4D Curves: Global Minimal 4D Paths to 3D Tubular Structure Extraction, IEEE MMBIA (2006).

[102] M. Niemeijer, J.J. Staal, B. van Ginneken, M. Loog, and M.D. Abramoff, Comparative Study of Retinal Vessel Segmentation Methods on a new Publicly Available Database, SPIE Medical Imaging (2004) 648-656.

[103] M. Poon, G. Hamarneh, and R. Abugharbieh, Segmentation of Complex Objects with Non-spherical Topologies from Volumetric Medical Images using 3D Livewire, SPIE Medical Imaging (2007) 1–10.

[104] M. Poon, G. Hamarneh, R. Abugharbieh, Live-Vessel: Extending Livewire for Simultaneous Extraction of Optimal Medial and Boundary Paths in Vascular Images, MICCAI (2007) 444-451.

[105] A. Clements, H. Zhang, Minimum Ratio Contours on Surface Meshes, IEEE SMI (2006) 57-78.

[106] J. O'Rourke, Art Gallery Theorems and Algorithms, Oxford University Press (1987).

[107] S. Konig, J. Hesser, 3D Live-Wires on Pre-Segmented Volume Data, SPIE Medical Imaging (2005) 1674-1681.

[108] A. Crauser, K.Mehlhorn, U. Meyer, and P. Sanders, A Parallelization of Dijkstra's Shortest Path Algorithm, MFCS (1998) 722-731.

[109] J. Jomier, V. LeDigarcher, and S. Aylward, Comparison of Vessel Segmentations using STAPLE, MICCAI (2005) 523-530.